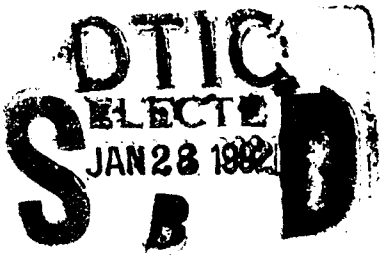
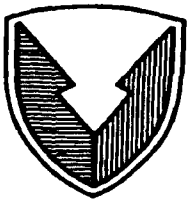
AD-A245 339

November 1991

TACTICAL COMMUNICATIONS NETWORK MODELLING AND RELIABILITY ANALYSIS: AN OVERVIEW



92-01947



US Army Laboratory Command
USA SURVIVABILITY MANAGEMENT OFFICE
Adelphi, MD 20783-1145

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

JC-2091-GT-F3

TACTICAL COMMUNICATIONS NETWORK MODELLING AND RELIABILITY ANALYSIS: AN OVERVIEW

November 1991

prepared by

J. S. Lee Associates, Inc.
Suite 609
451 Hungerford Drive
Rockville, Maryland 20850

under subcontract to

Defense Research Technologies, Inc.
354 Hungerford Drive
Rockville, Maryland 20850

for

Survivability Management Office
U. S. Army LABCOM
2800 Powder Mill Road
Adelphi, Maryland 20783

Contract DAAL02-89-C-0040

REPORT DOCUMENTATION PAGE

Form Approved
OPM No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of Management and Budget, Washington, DC 20503.

1 AGENCY USE ONLY (Leave Blank)	2 REPORT DATE	3 REPORT TYPE AND DATES COVERED
	NOVEMBER 1991	Final

4 TITLE AND SUBTITLE
Tactical Communication Network Modeling and Reliability Analysis: An Overview

5 FUNDING NUMBERS

6 AUTHOR(S)
James J. Kelleher

7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
J.S. Lee Associates Incorporated, Suite 609, 451 Hungerford Drive, Rockville, MD 20850

8 PERFORMING ORGANIZATION REPORT NUMBER
JC-2091-GT-F3

9 SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Survivability Management Office
Attn: SLCSM-C3I
2800 Powder Mill RD.
Adelphi, MD 20783-1145

10 SPONSORING/MONITORING AGENCY REPORT NUMBER

11 SUPPLEMENTARY NOTES

12a DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution is unlimited

12b DISTRIBUTION CODE

13 ABSTRACT (Maximum 200 words)

This report summarizes a survey of the literature on network reliability problems that arise in modeling communication networks and includes brief tutorials on the graph-theoretic concepts. Emphasis is placed on mobile communications in which the underlying network topology is subject to wide variations in the local environment, as opposed to network synthesis and design. The analysis of communication networks is primarily based on mathematical theory of graphs and its development of efficient algorithms to take advantage of parallel or concurrent data processing. The main thrust of graph theory in communications is the notion of the connectivity of the graph and this report concentrates on this subject area.

14 SUBJECT TERMS	15 NUMBER OF PAGES
Network Reliability Analysis, Tactical Communication Modeling, Connectivity, Graph Theory, Network Modeling	89
	16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18 SECURITY CLASSIFICATION OF THIS PAGE	19 SECURITY CLASSIFICATION OF ABSTRACT	20 LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

TACTICAL COMMUNICATIONS NETWORK MODELLING AND RELIABILITY ANALYSIS: AN OVERVIEW

1. Introduction 1

2. Graphs and Networks 5

2.1 Basic Definitions 5
2.2 Directed Graphs 7
2.3 Weighted Graphs 7
2.4 Graph Representations 8

3. Graph Connectivity 11

3.1 Components of a Graph 11
3.2 Measures of Connectivity 12
3.3 Classical Results on Connectivity 14
3.4 Connectivity in Directed Graphs 14
3.5 Trees 16

4. Graph Algorithms 19

4.1 Algorithm Complexity 19
4.2 NP-Hard and NP-Complete Problems 20
4.3 Graph Traversals 21
4.4 Examples 23
4.5 Other Graph Algorithms 29

5. Probabilistic Networks and Reliability 31

5.1 Modeling of Communications Networks 31
5.2 Reliability 32
5.3 Special Cases 33
5.4 Reliability Algorithms 34

DTIC COPY INSPECTED

Dist | Special
A-1

6. Calculation of Reliability 37

6.1 State Space Enumeration 37
6.2 Inclusion-exclusion Methods 39
6.3 Disjoint Sums 43
6.4 Dotson's Method 45
6.5 Factorization Methods 50
6.6 Approximations and Simulations 56

7. Other Performance Measures 61

7.1 Connectivity Factors 62
7.2 Network Diameter 64
7.3 Measures of Vulnerability 67

8. References 71

8.1 Graph Theory - General References 71
8.2 Algorithms for Graphs and Networks 71
8.3 Computational Complexity 73
8.4 NP-Hard and NP-Complete Problems 74
8.5 Dependent Failure Events 75
8.6 Network Reliability - Special Cases 75
8.7 Survey Articles and Bibliographies 76
8.8 Parallel Graph Algorithms 78
8.9 State Space Enumeration 78
8.10 Inclusion-Exclusion Methods 79
8.11 Disjoint Sums and Boolean Minimization 80
8.12 Dotson's Method 81
8.13 Factorization Methods 81
8.14 Bounds and Approximations 82
8.15 Simulations and Monte Carlo Methods 83
8.16 Connectivity Factors 84
8.17 Graph Diameter and Connectivity 85
8.18 Other Measures of Vulnerability 86

1. INTRODUCTION

This report summarizes a survey of the literature on network reliability problems that arise in modelling communication networks and includes brief tutorials on the pre-requisite graph-theoretic concepts. Emphasis is placed on mobile communication systems in which the underlying network topology is subject to wide variations in the local environment, as opposed to questions of network synthesis and design. For the latter problems, parts of the network configuration (such as the location of the transmitting and receiving nodes and the selection of the communication links) are at least to some extent free to be chosen by the system designer. In mobile command and control communication (C^3) systems this is not the case, as communication links may be limited to line-of-sight paths between nodes that are constantly changing position. Moreover, in a hostile environment the network topology is subject to link and node outages due to battle damage or enemy jamming.

The analysis of communication networks is primarily based on the mathematical theory of *graphs*, which has found wide applications to many different areas. This is a major branch of combinatorial mathematics that has been studied for centuries, and the research literature in the field is enormous. Recent activity in graph theory, however, is more intensive than ever. The emphasis is on the development of efficient algorithms, and many of the most important results here have been discovered only in the past decade. This aspect of the theory is now more important than ever because of the current importance of parallel or concurrent data processing.

The main aspect of graph theory in communications is the notion of the *connectivity* of a graph, and this report concentrates primarily on this aspect of the subject. For a graph to be connected the existence of a path from a given node to any other node is required, and this depends on the details of the node-to-node configuration that describes the graph. This is a completely *deterministic* question which can be quickly answered once the graph has been specified and represented in a form appropriate for processing in a digital computer. There are a number of efficient algorithms for determining graph connectivity, generally implemented by a systematic *traversal* of the graph beginning at a specified initial node. More quantitative versions describe how the graph can become disconnected due to the loss or deletion of some of its nodes or edges.

Of more importance in modern communication theory is the idea of a *probabilistic* network, which is simply a graph in which the individual nodes and/or edges are subject to failure. The general assumption here is that the failure probabilities are known and the individual component failures are statistically *independent.* (This second assumption is frequently unrealistic, but is necessary to simplify the mathematical analysis of the networks.) In this setting the underlying (deterministic) graph may itself be connected, but the unreliability of its components give rise to failure events for which connectivity is lost. It is the overall probability of such failures that measure the reliability of the network.

Unfortunately, the exact calculation of this type of unreliability measure is generally a computationally *intractable* problem and can be carried out only for networks of relatively limited size (e.g., 20-25 nodes). Over the past few years a number of approaches to this problem have been considered, and these methods are constantly being modified and improved. The most important of these are described, along with their relative advantages and disadvantages, but they are all subject to the computational difficulty mentioned above. That is, as the size of the network under consideration grows, the demands made on the available computational resources (run time or storage capability) become insurmountable. This seems to be inherent in the reliability problem itself, not a defect of the particular algorithm being applied to its solution.

While it is possible to restrict the node-edge configuration of the underlying graph to such an extent that efficient reliability algorithms do become available for such networks, the constraints involved are so severe as to preclude their use in modeling mobile radio networks. Thus future algorithmic improvements may allow the computation of reliability for ever larger networks, but the combinatorial complexity of the problem will eventually overwhelm the capability of any computing equipment being used.

For this reason a number of researches have been devoted to developing algorithms which would give reasonable bounds on network reliability. While these have met with some success, in general the intractability of the computation is still present and the usefulness of most of these methods to general network analysis is questionable. Another possibility is that of simulating the network success/failure events by Monte Carlo methods, although these have slow convergence properties. A more promising approach is the development of parallel processing and graph algorithms to be implemented on parallel computers. This area is quite new and much more work

remains to be done here.

There are many other graph-invariant properties that are closely related to questions of graph vulnerability or survivability, and a discussion of these makes up the concluding section. It is not clear at this point which of these are most applicable to mobile communication networks, although many have proved to be useful in other areas. These quantities are also the objects of much contemporary research, including the search for effective algorithms for their calculation.

$G = G(V, E)$	a graph (directed or undirected)
$V(G)$	set of nodes (vertices) of G
$E(G)$	set of edges (links) of G
$\mid S \mid$	number of elements of a set S
p	$\mid V(G) \mid$, number of nodes
q	$\mid E(G) \mid$, number of edges
K_p	complete graph on p vertices
$\deg(v)$	degree of vertex $v \in V(G)$
$\delta(G)$	minimum degree
$\Delta(G)$	maximum degree
$\mathrm{Diam}(G)$	diameter of G
$\mathrm{dist}(u, v)$	hop distance between nodes u and v
$\mathrm{Ngbd}(v)$	set of vertices adjacent to node u
$\kappa(G)$	vertex connectivity of G
$\lambda(G)$	edge-connectivity
$c(G)$	number of components of G
$G - e$	subgraph with edge e deleted
G/e	subgraph with edge e contracted
$G - v$	subgraph with node v deleted
$\lceil x \rceil$	smallest integer $n \geq x$ (ceiling)
$\lfloor x \rfloor$	largest integer $n \leq x$ (floor)

GLOSSARY OF NOTATION USED

2. GRAPHS AND NETWORKS

Communication networks are usually modelled by graphs or digraphs where the nodes or vertices of the graph are the transmitting/receiving facilities (possibly used simply as relays), while the edges of the graph indicate the possibility of communication between nodes. In this section we discuss the basic graph-theoretic concepts of most interest and indicate how these apply to modelling communication networks. The literature in graph theory is quite extensive and references [1-10] give a number of recent texts that introduce the field at various levels of sophistication. In particular, the book by Harary [7] has become a standard since its original publication in 1969.

The number of interesting quantities that are associated with a graph is *very* large, but we limit ourselves to those with immediate application to the possibility of communicating within the graph and to measuring its performance and reliability. We also indicate how graph data can be structured in a digital computer. There are many ways of doing this, some of which are more appropriate to mobile communication networks than others. Examples of these representations are given for a simple graph.

2.1 BASIC DEFINITIONS

A *graph* $G = G(V, E)$ consists of a set $V = V(G)$ of $p \geq 1$ *nodes*, or vertices, together with a set $E = E(G)$ of $q \geq 0$ unordered pairs of distinct nodes. We say that G has order p and size q and refer to G as a (p, q)-graph. The elements of E are the *edges*, or links, of G. An edge $e = \{u, v\}$ is said to join u and v in G, and we write this as $e = uv$. This definition excludes self-loops (an edge joining a node to itself) and parallel edges (two or more edges joining a pair of vertices).

The edge uv is said to be *incident to* the nodes u and v, which are *adjacent* nodes, or *neighbors*. Two edges are said to be adjacent if they are incident to a common node, a node is *isolated* if no edge is incident to it, and we define

$$\mathrm{Ngbd}(u) \triangleq \{v : uv \in E(G)\}, \tag{2.1}$$

the set of neighbors of node u. The *complete* graph on p nodes, denoted by K_p has $q = p(p-1)/2$ so that every pair of distinct nodes is adjacent.

The *degree* of a node $v \in V(G)$, $\deg(v)$, is the number of edges incident to it, and it

is always true that

$$\sum_{v \in V(G)} \deg(v) = 2q. \tag{2.2}$$

We also define the minimum and maximum degrees as

$$\delta(G) \triangleq \text{Min}\{deg(v):\ v \in V(G)\} \tag{2.3}$$
$$\Delta(G) \triangleq \text{Max}\{deg(v):v \in V(G)\}.$$

The graph is said to be *k-regular* if $\delta(G) = \Delta(G) = k$. (2.4)

Two graphs are *isomorphic* if there exists a one-one correspondence between their node sets that preserves adjacency. An *invariant* of a graph (G) is a quantity associated with G which has the same value for all graphs isomorphic to G. Thus p, q, δ, and Δ are graph invariants, as is the property of being k-regular, $0 \leq k \leq p$. Graphs are generally represented by point-line diagrams— the nodes given by distinct points and the presence of an edge joining two nodes indicated by a line segment or arc between them. Figure 2.1 shows such diagrams for a pair of isomorphic graphs. (The node correspondence is given by 1→2′ ,2→1′, 3→3′, 4→5′, and 5→4′.)

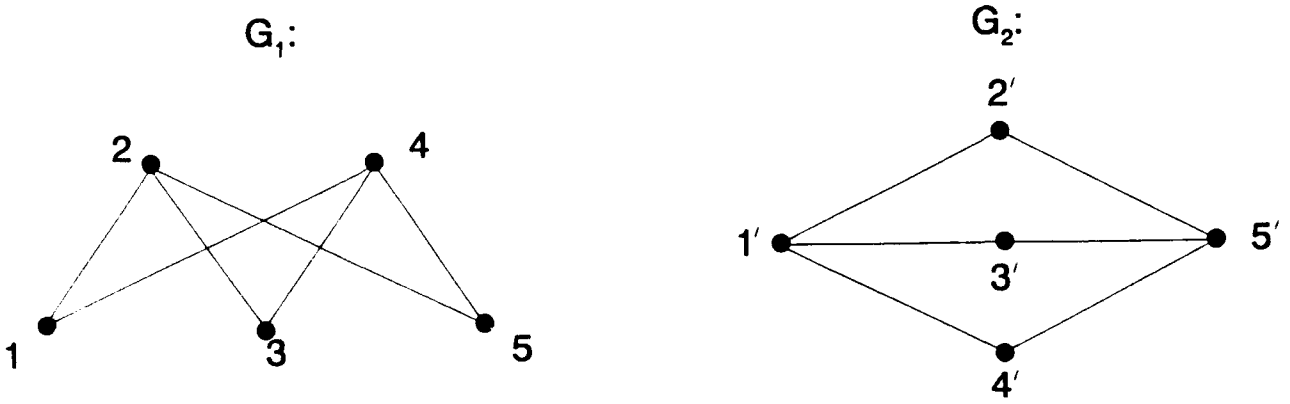


FIGURE 2.1 G_1 AND G_2 ARE ISOMORPHIC

A *subgraph* of G is a graph all of whose edges and nodes are edges and nodes, respectively, of G. It is a *spanning* subgraph if it contains all the nodes of G. Given a subset $V_0 \subset V(G)$ of nodes, the subgraph G_0 of G *generated* by V_0 is the largest subgraph of G with node set V_0. (That is, two nodes in V_0 are joined in G_0 if and only if they are joined in G.)

Also, for any node set $V_0 \subset V(G)$ we define $G - V_0$ to be the largest subgraph of G which contains no nodes belonging to V_0. (Note that it cannot contain any edges of

G which are incident to a node in V_0—if an edge belongs to a graph then the vertices which it joins must belong to the node set of that graph.) Similarly, for a set of edges $E_0 \subset E(G)$, $G - E_0$ denotes the largest subgraph of G which has no edges in E_0. Finally, since any (p, q)-graph G may be thought of as a subgraph of K_p, we can consider enlarging G by adjoining edges—this will give the smallest subgraph of K_p which contains G and the adjoined edges.

2.2 DIRECTED GRAPHS

A directed graph, or *digraph*, is defined exactly as a graph $G = G(V,E)$ except that the edges $E = E(G)$ now consist of *ordered* pairs of distinct nodes. We still write $e = (u,v) \in E(G)$ as $e = uv$, but it is now considered as a link *from* u *to* v. Its presence in $E(G)$ does *not* imply that the opposite link $e' = vu$ is also in $E(G)$. If this is always the case then we say that G is a *symmetric* digraph, and *asymmetric* if it is never the case. Associated with G is the *underlying* graph G_u of G obtained by ignoring the directions of the edges in the digraph.

Given a node $v \in V(G)$ we define the *indegree* of v to be the number of nodes $u \in V(G)$ for which $uv \in E(G)$. Similarly, the *outdegree* of v is the number of edges in $E(G)$ which go from v to some other node $u \in V(G)$. The degree of v is the sum of the indegree and outdegree, and we always have

$$\sum_{v \in V(G)} \text{indeg}(v) = \sum_{v \in V(G)} \text{outdeg}(v) = q. \tag{2.5}$$

The *complete* digraph on p nodes has $q = p(p-1)$ edges, one from each node to every other node. We will continue to denote it by K_p— it should be clear from the context whether we are referring to undirected or directed graphs.

Most of the definitions given previously for undirected graphs go over to directed graphs without change. The major differences between the two types are in their connectivity properties, which will be discussed later in this report. References [8] and [10] are devoted exclusively to digraphs, but all of the other texts cited for graph theory discuss them to some extent.

2.3 WEIGHTED GRAPHS

A weighted graph is simply a graph $G = G(V,E)$ together with a *weight function*

w: $E(G) \to \mathbb{R}$, where $w(e)$ is the weight assigned to the edge $e \in E(G)$. Typically we shall assume that $w(e) \geq 0$ for all edges e. In applications this might represent the cost of a link in a transportation network or the capacity of a communications channel in a radio network.

It is also possible to consider graphs with weights assigned to the nodes $v \in V(G)$, representing the costs associated with storage locations in a transportation network or the capacity of central processing nodes in a data network. In our applications we shall generally consider only functions defined on the edges of the graph.

Similarly, the definition of a weighted digraph is the same. The main point to note here is that the weight of an edge between two nodes will depend on the direction of the edge. Even if the digraph is assumed to be symmetric, this is *not* to be required of the weight function. That is, if $u, v \in V(G)$ are such that the edges $e = uv$ and $e' = vu$ both belong to $E(G)$, it is *not* necessarily true that $w(e) = w(e')$.

Thus, in a transportation system it is less expensive to move material downstream than to oppose the current in the opposite direction. Our main interest will eventually be in communication networks given by a symmetric digraph with a weight function $P: E(G) \to [0, 1]$. Here $P(e)$ represents the probability that the communication link $e \in E(G)$ is operational, and this probability must be allowed to depend on the direction of the link in order to model such things as local jamming, directional antennas for transmission and/or reception, etc.

2.4 GRAPH REPRESENTATIONS

To represent a graph in a computer the data must be organized in the machine so as to reflect the relations among the nodes $V(G)$ as specified by the collection of edges $E(G)$. One way to do this is to use the so-called *adjacency matrix* $A = A(G)$, a $p \times p$ matrix defined as

$$a_{ij} = \begin{cases} 1 & \text{if } i, j \in E(G) \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

Here the nodes have been mapped over to the set of integers $\{1, \ldots, p\}$. (That is, the i-th node read into the machine is assigned to the integer i.) This matrix has zeros on the main diagonal (no self-loops) and is symmetric if G is a graph or a symmetric digraph. (That is, $a_{ij} = a_{ji}$ for all $1 \leq i, j \leq q$, so only $q(q-1)/2$ matrix entries need to be stored.)

Other variations of this technique are possible. If it is desired to keep track of the edges, then the ij-entry of A would be the number of the edge $e = ij$ in the order in which it appeared in the input. Also, for a weighted graph the matrix could contain the weights of the corresponding edges. Graph-processing algorithms are discussed in more detail below and in references [11-28], but it is important to note that the way a graph is represented in the computer can have a large effect on the efficiency of an algorithm.

For example, one problem with the adjacency matrix representation is that if there are relatively few edges in the graph (as compared with the complete graph K_p), then most of the entries in A will be zero, thereby using a lot of unnecessary computer storage. In this case one would prefer to use the so-called *adjacency list* representation, where each vertex i is associated with a linked list containing the vertices which are adjacent to it. (For a digraph we would maintain two sets of lists, one for nodes adjacent to an edge from i and the other for nodes adjacent to an edge going to i.) For *dense* matrices, however, where the number of edges is close to that for K_p, the matrix representation is to be preferred.

Other considerations are also important. If we consider G as a subgraph of K_p and enlarge it by adjoining edges, or decrease it by removing some edges and/or nodes, it is much easier to reflect such changes in the adjacency matrix than to update the linked lists of vertex connections. However, the matrix representation can be costly in processing the graph data. In many cases we are given a vertex $v \in V(G)$ and are faced with the problem of selecting some subset of Ngbd(v), the neighboring nodes. The adjacency lists are clearly superior for such a purpose, whereas in the adjacency matrix we would have to scan across a whole row (or column) of A which may consist almost entirely of zeros.

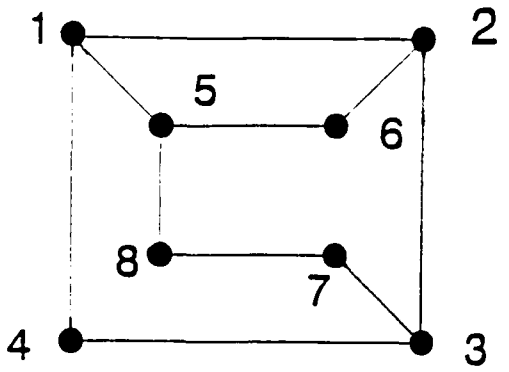
Nevertheless, we shall prefer to use the matrix representation. The main reason for this is our interest in modelling communication networks in widely varying environments. Under benign conditions with most nodes within line-of-sight of one another and with no hostile activity, almost all of the links will be operational and the linked list adjacencies have no storage advantage, but will exact a higher cost in overhead. Under jamming conditions and more stressful environments some links may drop out or be disabled, but the adjacency matrix is more easily updated to take account of such changes. The penalty for use of this representation is that one must use graph processing algorithms which are inefficient in dealing with sparse graphs. These various representations are shown for an example graph in Figure 2.2.

Example Graph:

8 Vertices
10 Edges (vs 28 edges in the complete graph)

Edge Listing:

1	2
1	4
1	5
2	3
2	6
3	4
3	7
5	6
5	8
7	8



Adjacency Matrix:

0	1	0	1	1	0	0	0
1	0	1	0	0	1	0	0
0	1	0	1	0	0	1	0
1	0	1	0	0	0	0	0
1	0	0	0	0	1	0	1
0	1	0	0	1	0	0	0
0	0	1	0	0	0	0	1
0	0	0	0	1	0	1	0

Adjacency Lists:

Node	Ngbd(Node)
1	2 4 5
2	1 3 6
3	2 4 7
4	1 3
5	1 6 8
6	2 5
7	3 8
8	5 7

FIGURE 2.2 GRAPH REPRESENTATIONS

3. GRAPH CONNECTIVITY

Reliability in communication networks is a measure of the ability of nodes of the network to establish communication with one another, either directly or through relay facilities. In particular, a communication path must exist between *any* two nodes of the network, for otherwise one part of the network would be isolated from another. This is determined by the *connectivity* of the underlying graph or digraph used to model the network. Moreover, we would like to have a large number of alternative paths between nodes so that the failure of a few network components would not result in such isolated nodes.

This section defines the basic notions of connectivity in graphs and digraphs and discusses some graph invariants to quantify network connectivity and vulnerability. Network reliability is a difficult problem because it is concerned with the overall *global* performance of the network, while the underlying graph is defined explicitly only in terms of the *local* interconnections among a large number of components. Connectivity is an extremely important property of a graph and any introduction to graphs will include some discussion of it. For its relationship to networks see the text by Colbourn [14], which is the best available reference to this subject.

3.1 COMPONENTS OF A GRAPH

A *walk* in a graph G is an alternating sequence of nodes and edges

$$v_0,\ e_1,\ v_1,\ e_2,\ \ldots,\ v_{n-1},\ e_n,\ v_n \tag{3.1}$$

where $e_k = v_{k-1}\, v_k$ for $1 \le k \le n$. It is said to *connect* v_0 with v_n and has *length* n. A walk is *closed* if $v_0 = v_n$, otherwise *open*. It is called a *trail* if its edges are distinct, and a *path* if its nodes (hence also its edges) are distinct. A *circuit* is a closed walk and a *cycle* is a circuit with distinct nodes (aside from the initial node $v_0 = v_n$). It is generally sufficient to denote a walk by its node sequence, the edges being implicit in the nodes.

Two vertices $u, v \in V(G)$ are said to be *connected in* G if there is a path from u to v in G, and G is said to be a *connected graph* if every pair of nodes is so connected. By convention each $v \in V(G)$ is connected to itself, so the property of being connected in G is an equivalence relation on $V(G)$. This relation therefore induces a partition of $V(G)$ into a number of disjoint subsets $V_1, \ldots, V_k$ whereby two nodes are connected in G if and only if they belong to the same subset of this partition. The number k of sets in the

partition is a graph invariant which will be denoted by $c(G)$, so G is connected if and only if $c(G) = 1$.

The subgraphs $G_1,\ldots,G_k$ of G generated by the $V_1,\ldots,V_k$ are referred to as the *components* of G, and these are the maximal connected subgraphs of G. These form a disjoint partition of G (all the nodes *and* edges of G are accounted for), and in many cases one can restrict one's attention to connected graphs only – otherwise consider its components in turn. The graph G is said to be *disconnected* if $c(G) > 1$ and *totally disconnected* if $c(G) = p$, whereby every $v \in V(G)$ is isolated in G.

If u and v belong to the same component of G, then we define the *distance* $\mathrm{dist}(u,v)$ between them as the length of the shortest path joining them. (We also set $\mathrm{dist}(u,u) = 0$, and $\mathrm{dist}(u,v) = +\infty$ if u and v belong to different components of G.) This induces a *metric* on each component of G, that is, it is symmetric and satisfies the triangle inequality

$$\mathrm{dist}(u,v) = \mathrm{dist}(v,u) \tag{3.2a}$$

$$\mathrm{dist}(u,w) \leq \mathrm{dist}(u,v) + \mathrm{dist}(v,w). \tag{3.2b}$$

A graph which is used to model a communications network is generally assumed to be initially connected. (Otherwise we study each component as a network by itself.) In this setting the metric defined above is usually referred to as the *hop distance*. If $(v_0, v_1,\ldots, v_n)$ is a path in G, we can imagine sending a message from node v_0 to node v_n through the $n-1$ relay nodes $v_1\ldots,v_{n-1}$. A total of n transmissions would be required by this path. Thus the hop distance $\mathrm{dist}(u,v)$ is the minimum number of transmissions required to realize communication between nodes u and v. Nodes in different components of the network cannot be so joined, and such communication is not possible.

3.2 MEASURES OF CONNECTIVITY

Obviously connectivity of a graph G is desirable for communications, so one would want to quantify this property and determine how "well-connected" it is. The most common way to do this is by specifying the number of nodes or edges that must be removed in order to disconnect G. For even more information we would want to identify those particular nodes or edges that should be removed in order to do this, that is, those to which the network is most vulnerable.

Thus, we say that $v \in V(G)$ is a *separating node* (or *cut-node*) of G if

$c(G - V) > c(G)$. Similarly, $e \in E(G)$ is a *separating edge* of $c(G - e) > c(G)$. A *non-separable* or *biconnected* graph is one that is connected and has no separating node, and a *block* of G is a maximal non-separable subgraph. Note that the blocks of a graph G are not necessarily disjoint so as to form a partition of $V(G)$, but two blocks can have at most one separating node of G in common. This type of connectivity can be characterized as follows—not only can any pair of vertices be joined, but this can be done through any specified edge.

More generally, we will say that a connected graph G is *n-connected*, $n \geq 1$, if it remains connected after the removal of any set of $n - 1$ vertices. The *node-connectivity* $\kappa(G)$ is the smallest number of vertices whose removal results in a disconnected graph. In a similar way, we define an *n-edge-connected* graph and the *edge-connectivity* $\lambda(G)$. It is always true that

$$\kappa(G) \leq \lambda(G) \leq \delta(G), \tag{3.3}$$

and this result is best-possible. (That is, given any three integers a, b, c for which $1 \leq a \leq b \leq c$, there exists a connected graph G for which $\kappa(G) = a$, $\lambda(G) = b$, and $\delta(G) = c$.) The book of Capobianco and Molluzzo [4] is a good reference for these inequalities and illustrates by example the fact that they cannot be improved.

For communication networks it is desirable to have large values of $\kappa(G)$ and $\lambda(G)$, as this provides for many alternative paths for communication between nodes. Conversely, small values of these graph invariants imply that the network is relatively vulnerable and can be disabled (i.e., disconnected) by the removal of only a few of its nodes or links.

One should note, however, that in general a graph is more susceptible to node removals than to a loss of edges. For example, the loss of the central node in a star network will result in the remaining node set being completely disconnected, but for an edge removal this type of behavior is not possible. That is, if $k = \lambda(G) \geq 2$, then no possible loss of k edges can result in a graph with more than two components. Moreover, the removal of an edge (as opposed to that of a vertex) cannot decrease the number of components of a disconnected graph, and in general

$$\kappa(G - v) \geq \kappa(G) - 1 \qquad \text{for all } v \in V(G) \tag{3.4a}$$

$$\lambda(G) \geq \lambda(G - e) \geq \lambda(G) - 1 \qquad \text{for all } e \in E(G). \tag{3.4b}$$

3.3 CLASSICAL RESULTS ON CONNECTIVITY

In general, for a given number p of nodes, a (p,q)-graph will have better connectivity for larger q. Many of the standard results in graph theory illustrate this point, such as

(i) If $q < p-1$, then G is disconnected (3.5a)

(ii) If $q > (p-1)(p-2)/2$, then G is connected. (3.5b)

These statements, so disparate as to be essentially useless, use only the global invariants p,q of the graph. More precise results can be stated in terms of the local structure of the graph at its nodes. Specifically [4],

(iii) If $\delta(G) \geq p/2$, then $\lambda(G) = \delta(G)$. (3.5c)

(iv) If $\delta(G) \geq p-2$, then $\kappa(G) = \delta(G)$ (3.5d)

(v) If $\delta(G) > (p+n-2)/2$, where $1 \leq n < p$, (3.5e)
then G is n-connected (i.e., $\kappa(G) \geq n$.)

The most important results along these lines are the theorems of Menger and their many variations [3, 7].

Menger's Theorems (Node/Edge versions)

(i) $\kappa(G) \geq n$ if and only if for each pair $u,v \in V(G)$ of distinct, non-adjacent nodes there exist at least n node-disjoint paths which connect u and v.

(ii) $\lambda(G) \geq n$ if and only if for each pair $u,v \in V(G)$ of distinct nodes there exist at least n edge-disjoint paths which connect u and v.

Observe the difference in the types of u-v connections referred to here. Two edge-disjoint paths between u and v need not be node-disjoint, for different edges may be incident to a common node. Node-disjoint paths, however, are necessarily also edge-disjoint. Thus there should be fewer node-disjoint paths than edge-disjoint paths between any pair of vertices, indicating once more the greater sensitivity of graph connectivity to node removal than to edge removal.

3.4 CONNECTIVITY IN DIRECTED GRAPHS

Most of our definitions for (undirected) graphs transfer directly to digraphs, except

what was previously called a walk (resp. path) is now referred to as a *directed walk* (resp. path) *from* one node *to* another. The major difference is in the concept of connectivity, which has several interpretations when the edge directions are accounted for. We shall say that $v \in V(G)$ is *reachable* from $u \in V(G)$ if there is a directed path in G from u to v. This is not an equivalence relation, however, since there may not be a directed path from v back to u. We can talk about the "components" of G with respect to this relation (i.e., maximal subgraphs), but they do not form a partition of the graph into disjoint pieces [8], as illustrated in Figure 3.1. Here node 3 is reachable from both node 1 and node 4, but neither one of these two is reachable from the other.

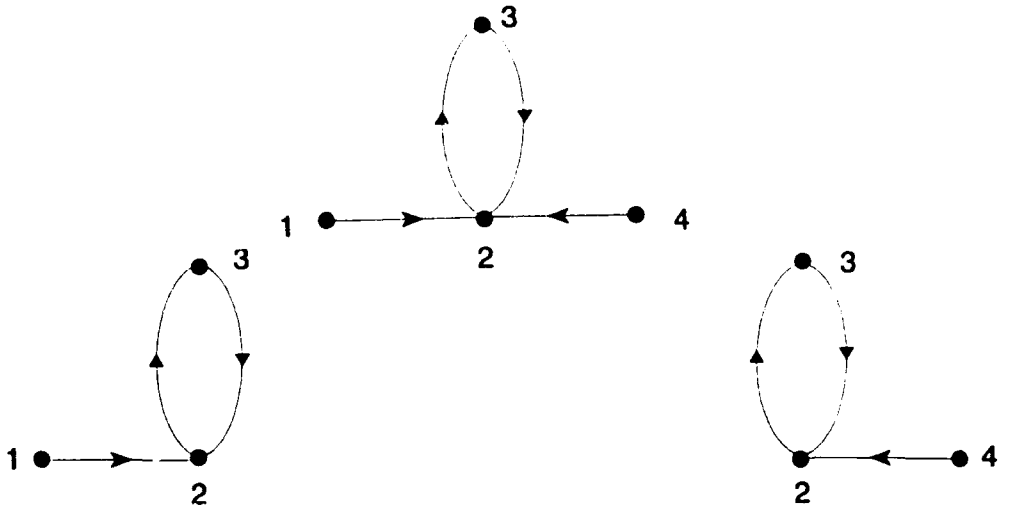


FIGURE 3.1 A DIGRAPH AND ITS "COMPONENTS"

The problem with this type of connectivity is that is does *not* result in a partition of the graph into disjoint subgraphs which can be analyzed separately. To avoid this difficulty we define two other types of connectedness. We shall say that $u, v \in V(G)$ are *weakly connected* if they are connected in the underlying graph G_u of G. This is simply the type of connectivity considered previously. We shall say that $u, v \in V(G)$ are *strongly connected* if each is reachable from the other. As this gives a true equivalence relation, we can talk about strongly (resp. weakly) connected digraphs and refer to the strong (resp. weak) components of digraphs..

There is still a problem here, illustrated by the first digraph shown in Figure 3.2—this digraph is weakly connected, but in the strong sense it is totally disconnected.

That is, each strong component is an isolated vertex, for in this digraph no vertex is strongly connected to any other.

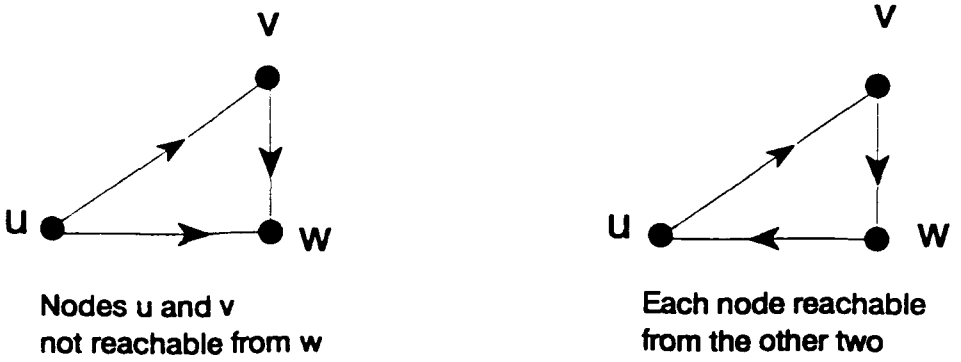


FIGURE 3.2 STRONGLY DISCONNECTED AND CONNECTED

Another type of problem is also shown in Figure 3.2, where the reversal of the direction of a single edge completely disconnects a strongly connected graph. We shall avoid these difficulties in what follows by usually assuming a *symmetric* digraph when modeling a communications network. In this case the reversal of any edge is still an edge, so the reversal of any path in G is again a path in G. For such graphs reachability is a symmetric relation and the notions of strong and weak connectivity coincide so that all of the previous discussion applies with only minor modifications.

This is admittedly a very strong assumption and would not ordinarily be allowed, for in many of the applications of digraphs the edge directions are of critical importance to the problems being modelled. In scheduling problems, for example, it is clearly impossible to execute one task before doing all those others which must feed into it. We can justify our assumption here only because we will eventually limit ourselves to stochastic networks in which each edge is assigned a probability of its being operational. From the probabilistic point of view then, the presence of any particular edge in a digraph is not important since it can be effectively eliminated from consideration by assigning it an operational probability of zero.

3.5 TREES

The most important type of connected graph is a *tree*, namely a (p,q)-graph T which is connected and for which $q = p - 1$. Such a graph is necessarily circuit-free and will be disconnected by the removal of any edge. Moreover, between any two

nodes of T there is a *unique* path joining them. A subgraph T of a connected graph G is a *spanning tree* if it is a spanning subgraph of G which is also a tree. The addition to T of any node or edge of G will then create a cycle in T. A node $v \in V(T)$ is *pendant* if $\deg(T) = 1$, while an edge $e \in E(T)$ is a *leaf* if it is incident to a pendant node.

If G is a weighted graph we say that a spanning tree T of G is *minimal* if

$$w(T) = \sum_{e \in E(T)} w(e) \tag{3.6}$$

is a minimum over all such spanning trees. Such spanning trees always exist, and can be constructed by means of the following [18].

Theorem: Let $v_0 \in V(G)$ and let $e_0 \in E(G)$ be such that $w(e_0) \leq w(e)$ for all edges e incident to v_0. Then there is a minimal spanning tree of G which contains the edge e_0.

This result is the basis for algorithms to construct a minimal spanning tree for any connected, weighted graph G starting from a given initial root node v_0. One well-known method is Prim's method [20], which may be described as follows:

1. Set $V_0 = \{v_0\}$, $T = 0$.
2. If $V_0 = V(G)$ then stop
3. Let $E_0 = \{e \in E(G)\colon e = uv,\ u \in V_0,\ v \in V(G) - V_0\}$
   $L = \min\{w(e)\colon e \in E_0\}$
   and choose $e \in E_0$ with $w(e_0) = L$, say $e = uv$
4. Let $v_0 = v_0 \cup \{v\}$
   $T = T \cup \{e\}$
   Go to 2

At any stage we adjoin to T the least expensive (i.e., smallest weight) edge which maintains the tree connectivity without forming a circuit. If we do not insist on keeping T connected we obtain Kruksal's method, which builds up a collection of disjoint trees which gradually merge into one another to exhaust $V(G)$. Also, for a disconnected graph this algorithm can be used to find the components of G, if in step 3 we choose $e \in E_0$ arbitrarily, but stop as soon as $E_0 = 0$. The set V_0 is then the set of nodes in the component of G which contains v_0. (The same procedure finds a spanning tree in an unweighted graph—simply define a weight function as $w(e) = 1$, $e \in E(G)$.)

Prim's method is an example of a *greedy algorithm* [21], which always chooses the best edge available at that time. That is, it makes a locally optimal selection, which for this example does in fact result in a global optimum, a minimal spanning tree. Greedy methods do not always yield optimal solutions, but for many problems in graph theory they do.

If T is a digraph we generally refer to a *rooted tree*, which has a distinguished node $v_0 \in V(T)$, the root, such that

(i) $\text{indeg}(v_0) = 0$ (3.7a)

(ii) $\text{indeg}(v) = 1$ for all $v \in V(T) - v_0$, (3.7b)

(iii) Every $v \in V(T)$ is reachable from v_0. (3.7c)

When we speak of a subtree or spanning tree of a digraph G we implicitly mean a subgraph of this type. Most of the above definitions make sense in this context and the graph algorithms used will be applied here. Note however, that Prim's algorithm gives a minimal spanning tree rooted at v_0. This is not necessarily a minimal spanning tree of G, for it is not known *a priori* how the root v_o should be chosen so as to give the desired global minimum.

4. GRAPH ALGORITHMS

Many problems are naturally formulated in terms of graphs or digraphs, such as transportation or communication networks, job scheduling in manufacturing, physical connectivity in electrical circuits, etc. Obvious questions arise, such as which nodes are connected to a given node, what path between two nodes is the least expensive, or when each manufacturing task should be scheduled.

For probabilistic graphs we don't ask if the graph is connected, but what the probability of that event may be, or we might want the average distance (expected value) between two vertices. We consider here some deterministic problems which process only the graph data $V(G)$ and $E(G)$, along with any edge costs if we have a weighted graph. Problems involving probabilistic networks will be considered below in Section 5, but there is a significant difference here. As we shall see below, it is very easy to determine if a graph is connected, but very difficult to compute the probability that this is the case.

Before answering these probabilistic questions we must first deal directly with the nodes and links of the graph that represents the network. For example, the probability of two nodes being connected depends on the existence of paths between them, so we would need to have an algorithm which returned a list of all such paths, if any. The area of graph algorithms is a very important part of graph theory, particularly for reliability problems which are computationally rather expensive. We are interested in efficient algorithms, where efficiency is judged by being able to run the algorithm to completion in an acceptably short time. Questions regarding available storage or internal/external memory requirements are not considered.

4.1 ALGORITHM COMPLEXITY

The complexity of an algorithm is generally measured by the runtime of its implementation on a digital computer versus the size of the input data used by the algorithm. It is important to recognize that this is a measure of the algorithm's *worst-case* performance. That is, we seek an *upper bound* $f(n)$ on the maximum number of elementary operations to carry out the algorithm as a function of the size n of the input data. It may happen that some inputs are processed very quickly, or that the average performance is fast, but we want a guaranteed upper bound for the computation over

all allowable inputs. References [29-38] are all good references to the general theory of algorithms and algorithmic complexity. Of these the book of Harel [33] is especially recommended as a very readable and wide-ranging view of the field, and Sedgewick [37] has a good discussion of algorithm implementation.

In our case the input data is a graph or digraph $G(V, E)$ consisting of p vertices and q edges, so the complexity of an algorithm will generally be estimated by a function of p and/or q. We shall use the O-notation here—given two functions $f(n)$ and $g(n)$, to say that $f(n) = O[g(n)]$ if there is a constant $M > 0$ so that $f(n) \leq Mg(n)$ for all sufficiently large n. The meaning of this is that $f(n)$ is no worse that $g(n)$, up to a multiplicative constant. An algorithm is said to run in polynomial time if its complexity satisfies $f(n) = O(n^k)$ for some $k \geq 0$. Special cases are constant time ($k = 0$) and linear time ($k = 1$). As an example, the well-known quicksort algorithm to sort a collection of n objects is of type $O(n^2)$, although its average performance is known to be $O(n \log n)$, much faster for large n.

Polynomial time is generally taken as the boundary between practical and intractable problems. Those problems for which there are no known polynomial-time algorithms must generally be solved using algorithms which exhibit exponential growth in the input size and are used only for relatively small networks. For faster computation one must settle for only an approximate solution, or restrict the input data to special networks which do admit a polymonial-time solution.

4.2 NP-HARD AND NP-COMPLETE PROBLEMS

Of special interest here are the so-called NP (non-deterministic polynomial) problems. Roughly speaking, these are problems for which there exists a polynomial-time algorithm which can verify the validity of any proposed candidate solution to the problem, although the problem itself may not be known to have a polynomial time solution. A problem is called NP-hard if it is at least as difficult as any NP-problem (that is, a polynomial-time solution for that problem would imply such a solution to *every* NP problem). An NP-hard problem is said to be NP-complete if it is itself an NP problem.

It is not known at this time if there exists *any* NP-problem which does *not* have a polynomial-time solution. This would require a *lower* bound on such a problem's complexity which grew faster than any polynomial function. The book of Garey and Johnson [32] is the most complete introduction to NP-completeness, although its

compilation of NP-complete problems is already quite dated.

Unfortunately, many well-known problems in graph theory have been shown to be NP-hard, including some that are related to network reliability [40, 41]. As an example, we consider the following two questions for a connected graph G.

(i) an *Eulerian tour* of G is a circuit of G in which each edge occurs exactly once.

(ii) a *Hamiltonian tour* of G is a spanning cycle of G. (This is closely related to the well-known traveling salesman problem.)

The question is whether such tours exist, but the answers are quite different. The Eulerian tour problem is solvable by an $O(p^2)$ algorithm, while the Hamiltonian tour problem is NP-complete, and as such the question is essentially unanswerable for large networks. The inefficiency of a deterministic algorithm for an NP-complete problem is due to the exponentially large number of possible solutions that must be dealt with, and it may be possible to decrease this expense by better defining and limiting the number and type of potential candidates in the search. While still resulting in exponential complexity, the order of growth is decreased and larger graphs can be processed in a prescribed amount of time. Because of its practical importance, the Hamiltonian tour problem has been more intensively investigated than any other NP-complete problem [3, 13]. For large graphs, however, all of the available search methods are intractable, and one must settle for approximate solutions [21].

It should be noted, however, that some problems in the theory of graphs are intrinsically non-polynomial in nature. For example, the number of minimal spanning trees in a completely connected graph on p vertices is p^{p-2}. Thus any algorithm to compute all the spanning trees of a graph must expect to work at least this hard, which is much worse than any exponential growth. Similarly the number of minimal cutsets and minimal paths in graphs also exhibit such explosive growth and cannot be completely specified by any polynomial-time algorithm. No NP-problem is known to be of this type.

4.3 GRAPH TRAVERSALS

A *traversal* of a connected graph is simply a procedure to visit the nodes of G in some systematic way, starting from some given node and moving to adjacent nodes via the edges of the graph. The two most common methods for doing this are the *depth-first* search and *breadth-first* search [20, 21]. Both can be used to construct spanning

trees of G and to determine the component of the graph that contains the initial root node.

In a depth-first search, immediately on visiting one node we move on to visit one of its unvisited neighbors, if any. If this is not possible we backtrack to the last node that was visited which does have unvisited neighbors and repeat the process. The procedure terminates only when the visited nodes have no unvisited neighbors, exhausting $V(G)$ if that graph was connected. This very simple procedure is inherently recursive and is best implemented in this way:

Depth-First Search

1. Given root node v_0, set $T = \emptyset$
2. Set $c(v) = 0$ for all $v \in V(G)$
3. Visit (v_0)
4. Return T

The initially empty edge set T will be a tree which is rooted at v_0 and spans the component of G containing v_0. Hence, for a connected graph, T is a spanning tree of G. It is constructed by successive calls to the recursive function Visit, that is

```
Visit(v)
    c(v) = 1
    For all u ∈ Ngbd(v) do
        If c(u) = 0 then
            Visit (u)
            T = T ∪ {uv}
```

In this function, each call to the function immediately marks the current vertex v as having been visited ($c(v) = 1$), but no edges are added to the tree T as long as that vertex still has neighboring vertices which have not yet been so marked. Instead, it moves to one of the unmarked neighbors and repeats the process. The function call finally returns with an edge added to T only when it has reached and marked as visited a vertex all of whose neighbors have already been visited. Thus the search is forced to look for new nodes to visit which are farther away from the root, and it will move backward towards the root only when it encounters a dead end (although it can be expected to form very long paths which will loop back closer to the root). It tends to

produce very long trees with few branches, as the *last* vertex processed yields the *first* edge of T.

Breadth-first search, on the other hand, visits *all* neighboring vertices of a given vertex before moving farther away from the root, and as such it tends to produce short, fat trees with many branches. It can also be given in recursive form, but is perhaps more naturally non-recursive. One version of breadth-first search is described as follows:

Breadth-First Search

1. Set $c(v) = 0$ for all $v \in V(G)$
   $i = 1$
   $c(v_0) = 1$
2. Let $S = \{v: c(v) = 0, u \in Ngbd(v)$ for some u with $c(u) = i\}$
   If $S = \emptyset$ then stop
3. $i = i + 1$
   $c(v) = i$ for all $v \in S$
4. Go to 2

This implementation of the breadth-first-search algorithm includes a vertex-labelling scheme which indicates how far the search has proceeded when it is visiting any particular vertex. (That is, it computes $c(v) = 1 + \text{dist}(v, v_0)$ for all vertices $v \in V(G)$ which are connected to the root vertex v_0.) For any given distance d from the root, the search will visit all nodes at distance d before moving out to look for nodes connected to v_0 at a distance of $d + 1$. Edges closer to the root are considered earlier in the process.

These procedures have many extensions and variations. Depth-first search can be used to check for biconnectivity [20], while the idea behind breadth-first search can be used to find minimal paths in weighted graphs [31]. The precise form of the tree constructed depends on the order in which the edges are represented. Also, if the graph were not connected then both of these algorithms visit those nodes of G which lie in the connected component containing the root node v_0. Thus these algorithms can check for the connectivity of G, or can find all of its components should it be disconnected.

4.4 EXAMPLES

As an example, in Figures 4-1 and 4-2 below, we show how the depth-first and

breadth-first methods might be implemented in Pascal when using an adjacency matrix representation of the graph. The algorithm outputs given in Figure 4-3 and diagrammed in Figure 4-4 show their operation on the (8, 20)-graph whose edge list is shown. The edges, starting from node #1, are shown in the order processed by the computer. Thus in the depth-first-search algorithm the edge from node 3 to node 4 is the first edge added, since node 4 is the first vertex from which the search has to backtrack from a dead end. Conversely, the edge from node 1 to node 2 is the last edge added to the tree, although it was the first link traversed by the algorithm. The breadth-first-search, however, also traversed this link first but adjoined it to the tree immediately. Note that these two algorithms are both of complexity $O(p^2)$, due to the use of loops for the adjacency matrix representation. For sparse matrices the use of adjacency lists can do much better, say $O(p+q)$.

```
CONST
  MAXV = 20;            (* Maximum number of graph nodes *)

TYPE
  NodeSet = Set OF 1..MAXV;
  NodeList = Array[1..MAXV] OF Integer;
  AdjMat = Array[1..MAXV] OF NodeList;

VAR
  A : AdjMat;           (* A = the adjacency matrix of the graph  *)
  NodeNum : Integer;    (* NodeNum = number of nodes in the graph *)

(*-----------------------------------------------------------------*)
PROCEDURE DFS(Root : Integer);
(* Depth-First Search of graph from root node *)
VAR
  i : Integer;
  Tag : NodeList;
  (*-------------------------------------*)
  PROCEDURE Visit(k : Integer);
  (* Recursive traversal from node k *)
  VAR
    i : Integer;
  BEGIN
    Tag[k] := 1;
    FOR i := 1 TO NodeNum DO
      IF (A[i,k] <> 0) AND (Tag[i] = 0) THEN
        BEGIN
          Visit(i);
          Writeln('Edge: ', k, ' --> ', i);
        END;
  END; (* Visit *)
  (*-------------------------------------*)
BEGIN (* Start of DFS *)
  FOR i := 1 TO NodeNum DO
    Tag[i] := 0;
  Visit(Root);
END; (* DFS *)
```

NOTES: This is a recursive version of the depth-first search. The Tag vector is simply a Boolean labelling of the nodes, where Tag[v] = 0 if and only if node v has not yet been visited. The edges are written out in the reverse order of that in which they are traversed. Bookkeeping is kept to a minimum in this version of the algorithm.

FIGURE 4-1 DEPTH-FIRST SEARCH ALGORITHM

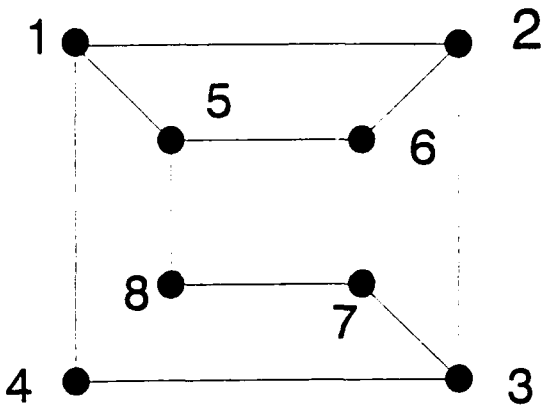
```
CONST
  MAXV = 20;            (* Maximum number of graph nodes *)

TYPE
  NodeSet = Set OF 1..MAXV;
  NodeList = Array[1..MAXV] OF Integer;
  AdjMat = Array[1..MAXV] OF NodeList;

VAR
  A : AdjMat;           (* A = the adjacency matrix of the graph  *)
  NodeNum : Integer;    (* NodeNum = number of nodes in the graph *)

(*-----------------------------------------------------------------*)
PROCEDURE BFS(Root : Integer);
(* Breadth-First Search of graph from root node *)
VAR
  Tag : NodeList;
  Next : NodeSet;
  i, j, Count : Integer;
BEGIN
  Count := 1;
  FOR i := 1 TO NodeNum DO
    Tag[i] := 0;
  Tag[Root] := Count;
  WHILE TRUE DO
    BEGIN
      Next := [];
      FOR j := 1 TO NodeNum DO
      FOR i := 1 TO NodeNum DO
        BEGIN
          IF (Tag[i] = 0) AND (Tag[j] = Count) AND (A[i,j] <> 0) THEN
            BEGIN
              Next := Next + [i];
              Tag[i] := Count + 1;
              Writeln('Edge:  ', j, ' --> ', i);
            END;
        END;
      IF (Next = []) THEN Exit;
      Inc(Count);
    END; (* WHILE *)
END; (* BFS *)
```

NOTES: This is a non-recursive version of the breadth-first search. The Tag vector here will label the vertices according to their distance from the initial node, where again Tag[v] = 0 implies that node v has not yet been visited. If the NodeSets Next were accumulated, then the final result would contain those nodes in the connected component of G which contains the root node.

FIGURE 4-2 BREADTH-FIRST SEARCH ALGORITHM

Example Graph:

Root Node: 1
No. of Nodes: 8
No. of Links: 20



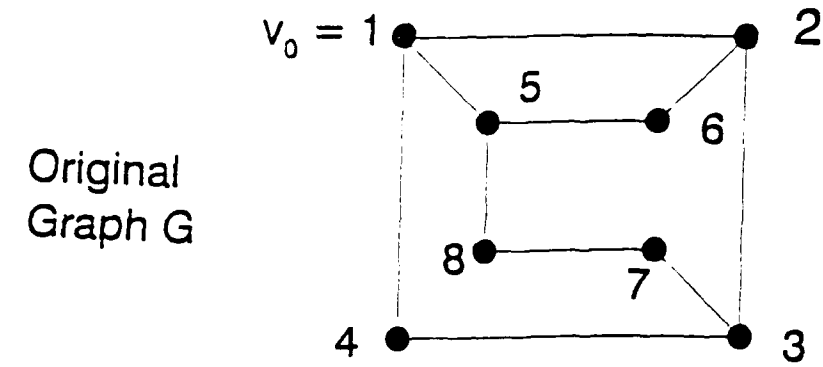
Edge Listings:

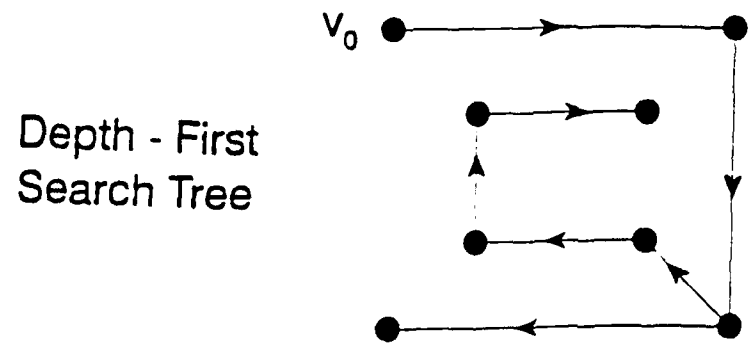
1 2
1 4
1 5
2 1
2 3
2 6
3 2
3 4
3 7
4 1
4 3
5 1
5 6
5 8
6 2
6 5
7 3
7 8
8 5
8 7

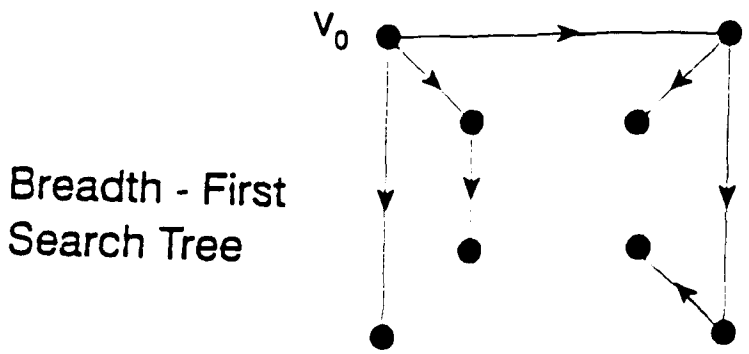
Algorithm Outputs:

DFS: Root = #1	BFS: Root = #1
Edge: 3 --> 4	Edge: 1 --> 2
Edge: 5 --> 6	Edge: 1 --> 4
Edge: 8 --> 5	Edge: 1 --> 5
Edge: 7 --> 8	Edge: 2 --> 3
Edge: 3 --> 7	Edge: 2 --> 6
Edge: 2 --> 3	Edge: 5 --> 8
Edge: 1 --> 2	Edge: 3 --> 7

FIGURE 4-3 EXAMPLE OUTPUTS OF SEARCH ALGORITHMS







NOTE: Max $\{\text{dist}(u,v_0): u \in V(G)\} = 3$ in the original graph and in the breadth-first search, but this value has increased to 6 with depth-first searching

FIGURE 4.4 GRAPH TRAVERSALS AND SPANNING TREES

4.5 OTHER GRAPH ALGORITHMS

Other important polynomial-time graph algorithms are those of Prim and Kruksal for constructing minimal spanning trees of a connected graph starting from an initial root node. Again, both of these algorithms are $O(p^2)$ when the graph is represented by an adjacency matrix. Also, shortest-path algorithms are basically breadth-first searches (as is Prim's algorithm) and give constructions such as Dijkstra's method and shortest-path spanning trees.

Transitivity problems also arise here. The transitive closure $G^* = G(V, E^*)$ of G is the graph for which $e = uv \in E^*$ if and only if there is a path from u to v in G. Here Warshall's method (compute the adjacency matrix of G^*, given that of G) and Floyd's algorithm (calculate the minimal cost of each $e \in E^*$) are used, both of which are of order $O(p^3)$.

Other NP-complete graph problems include longest path problems, minimum vertex colorings, maximal clique size (largest completely connected subgraph), and minimum vertex coverings (smallest set of nodes to which every edge is incident). No polynomial time algorithms are known for any of these questions, and the most commonly accepted opinion is that no such algorithms exist. That is, the NP-complete problems probably have no polynomial time solutions.

For NP-complete problems a polynomial-time algorithm can now be found only when some restrictions are made on the inputs allowed. For example, if a graph G satisfies

$$\deg(u) + \deg(v) \geq p \tag{4.1}$$

for all non-adjacent nodes $u, v \in V(G)$, then it is known that G has a Hamiltonian cycle. In this case there is an $O(p^2)$ algorithm which will find such a cycle, but this will fail for non-Hamiltonian graphs or for some Hamiltonian graphs which do not satisfy the above condition (which is *not* a necessary condition for Hamiltonicity). See Buckley and Harary [3] for a description of such an algorithm.

5. PROBABILISTIC NETWORKS AND RELIABILITY

Communication networks are generally modelled by weighted digraphs in which the nodes represent communication centers (transmitters, receivers, relays) and the edges are communication channels between the nodes. The weight associated with each edge is taken to be the probability of that edge being operational. A *reliability measure* on such a network is then the expected value (in the probabilistic sense) of some aspect of the connectivity of the underlying graph. In particular, we are interested in the probability that a path exists between two given nodes, or the probability that the network is connected [14].

This section describes some of the reliability measures of interest and discusses the problems involved in calculating them. Specific methods used for reliability computations are considered in the following section. Unfortunately, most problems in reliability are NP-hard [39-47]. While it is possible to obtain polynomial time algorithms for reliability calculations of some restricted classes of networks, such restrictions are far too strong to be applied to mobile communication networks operating in stressful environments.

5.1 MODELING OF COMMUNICATION NETWORKS

We shall take as our model of communication networks a weighted digraph $G = G(V,E)$ with the weight function $P : E(G) \rightarrow \mathbb{R}$ interpreted as the probability of the edges being operational. (That is, $1 - P(e)$ is the probability that edge $e \in E(G)$ has failed.) It is *not*, in general, assumed that $P(e) = P(e')$ for $e \in E(G)$, where $e' \in E(G)$ is the reversal of e. If this is always the case then we will refer to G as a *symmetric* or *undirected* network. Many of the results in the literature apply only to this case, which does *not* model some local node/link behavior, such as directional antennas for transmission and reception.

In what follows we shall concern ourselves primarily with link failures, which has been the situation dealt with most extensively in the literature. That is, we initially assume that the nodes are completely reliable, so we do not consider a corresponding probability function defined on the set of vertices of G. The problems posed by node failures, however, are of importance in mobile communications networks, so we will

return to this question below. Of most significance here is how some of the methods which have been developed to treat only edge failures can easily be modified to take into account the possibility of node failures as well.

An important assumption usually made at this point is that all edge failures (and node failures, if these are included in the model), are statistically *independent.* This assumption greatly simplifies the reliability calculation, but is unrealistic in many situations, such as when some communication equipment is shared by several links. However, the modeling of dependent equipment failures generally requires the use of conditional and/or joint probability distributions, of which there are potentially an exponentially large number [48-56]. Since most reliability problems are known to be $\mathcal{NP}$-hard, even in the enumerative sense, this greatly increases the difficulty of the computation, and we will assume independent component failures in what follows.

5.2 RELIABILITY

As described previously, network reliability is a measure, now in the probabilistic sense, of the connectivity of a network. The most common statistic used is the so-called *K-terminal reliability* where $K \subset V(G)$ is a distinguished subset of the vertices of the graph. The reliability problem is to calculate the probability that every two vertices in K are connected by a path of operational edges. Of particular interest here is the 2-terminal reliability, when $|K| = 2$, and the all-terminal reliability, where $K = V(G)$.

For 2-terminal reliability we have $K = \{s, t\}$ where $s, t \in V(G)$ are referred to as the *source* and *terminal* (or *sink*) respectively. The *st*-reliability of the network is the probability of the source being able to communicate to the sink. Note that if we have a non-symmetric network it is important which node is the source and which is the sink, as we are asking for the probability of a working path *from* the source *to* the sink. The *st*-reliability for an undirected network can be considered as a special case of that for a directed network (each edge is equivalent to two edges with opposite directions but *equal* probability of failure). The *st*-reliability will be denoted by $\mathrm{Rel}(G; s, t)$.

For an undirected network the all-terminal reliability is simply the probability that the network is connected, or, what is the same, the probability that the graph G contains at least one operational spanning tree or subgraph. For directed networks the all-terminal reliability measures the probability that the network is strongly connected. It seems, however, that strong connectivity, even in the probabilistic sense, is too restrictive a requirement. For example, using local jamming it might be relatively easy

to isolate a particular node on the periphery of a graph so that it could no longer receive messages from the rest of the network. Even if transmissions *from* this node could be relayed throughout the network to all the other nodes, the reliability of the network in the strong sense would be *zero*. This should be considered too pessimistic a reliability measure for the overall performance of such a network. The all-terminal reliability will be denoted by $\mathrm{Rel}(G)$, and for an arbitrary $K \subset V(G)$ we let $\mathrm{Rel}(G;K)$ denote the K-reliability of G.

Since we cannot abandon the use of directed networks if we want to model any direction-dependent behaviour in the network links, the all-terminal reliability is not an appropriate statistic. Instead we shall concentrate on the 2-terminal st-reliability. If a global figure for the network as a whole is required, we could use the average of these reliabilities over all possible $p(p-1)$ node pairs.

$$\mathrm{Res}(G) = \frac{1}{p(p-1)} \sum_{s,t \in V(G)} \mathrm{Rel}(G;s,t), \tag{5.1}$$

the so-called *resiliency* [63] of G. This measures the fraction of all possible source-sink pairs for which communication paths can be established. (Of course, we have $\mathrm{Rel}(G;s,t) = \mathrm{Rel}(G;t,s)$ if G is a symmetric network.)

5.3 SPECIAL CASES

It is an unfortunate fact of life that the calculation of any of these reliability measures for an arbitrary communications network is an NP-hard problem. This remains true even if a number of additional simplifying assumptions are imposed. For example, one might (unrealistically) require that all link probabilities are equal, say $P(e) = \beta_0$ for all $e \in E(G)$, where $0 < \beta_0 < 1$ is a constant. The reliability calculations can then be reduced to problems in graph enumeration. Let M_k be the number of paths from s to t of length k, $0 < k \leq q$. Then we have

$$\mathrm{Rel}(G;s,t) = \sum_{k \geq 1} M_k \beta_0^k (1-\beta_0)^{q-k} \tag{5.2}$$

Analogously, if N_k is the number of spanning subgraphs of G having k edges, then

$$\mathrm{Rel}(G) = \sum_{k \geq 1} N_k \beta_0^k (1-\beta_0)^{q-k} \tag{5.3}$$

However, even the graph enumeration questions of determining the coefficients M_k and N_k are NP-hard [47].

An alternative simplification is to restrict the class of networks under consideration [57-67], say to those whose underlying graph is *planar*. (That is, the nodes and edges can be drawn in the xy-plane in such a way that no two edges intersect, except at a node of the graph.) However, the reliability problem is also $\mathcal{NP}$-hard for planar networks [58]. This is even true for *grid networks* where the nodes are located at the points of a regular square lattice in the xy-plane and each node has direct edge connections only with its nearest neighbors [62]. While it is possible to still further restrict the networks being considered so as to obtain a class (e.g., series-parallel graphs) of graph inputs for which polynomial time reliability algorithms do exist, such restrictions are too severe for our purposes.

Indeed, for mobile communication networks it is not realistic to restrict the possible topology or node interconnectivity in any way whatsoever. The links in such networks are constantly changing as the nodes move in and out of range of one another. Moreover, under battlefield stress conditions, such as local jamming or attack, any link or node is subject to failure. In principle, then *any* (p,q)-digraph could occur as the underlying graph of a communications network. For the same reasons, no *a priori* restrictions could be placed on the values of the edge failure probabilities.

5.4 RELIABILITY ALGORITHMS

Thus any algorithms to be considered for use in our network reliability calculations should be able to deal with a completely general network with no constraints imposed on either the available edge connections or on the values of the link probabilities. A number of such methods have been developed in recent years, and improvements are constantly being made. We will discuss a few of these in some detail in the next section, but the reader should take note of the various survey articles on network reliability that have been previously published [68-82].

Not all of these bear directly on communications, some being either too specialized or emphasizing some aspect of reliability of less immediate interest to us, but all are worth perusing if only to get a different point of view. Thus [80] is concerned almost exclusively with planar networks, much too restrictive a condition to be required of a mobile communications network. Another example is [69], which stresses the applications of reliability criteria in the synthesis of communication networks. From this point of view the node/edge configuration of the graph is, at least to some extent, at the disposal of the designer and can be adapted as needed to satisfy minimal

reliability or vulnerability requirements.

One topic that has not been mentioned yet but which will obviously be of great interest in the future is the application of *parallel processing* techniques to solving graph-theoretic problems. Here the computer consists of many separate processing elements, each working simultaneously on its own part of the problem. With a large number of processors the time required to complete the computation will generally be much less than when the data is processed serially in the usual way through a single central processor.

For example, to simply add up N numbers obviously requires $N-1$ additions, hence $O(N)$ machine cycles on such a computer. However, if there were available N parallel processors, one for each item of data, then this can be done in $O(\log_2 N)$ cycles, an enormous saving for large N. Similarly, no all-purpose serial sorting algorithm can process N items in better than $O(N \log N)$ time, but much better algorithms can be devised to do this if N or more processors are available. Another example is that of matrix multiplication—if N^2 processing elements are available, then two $N \times N$ matrices can be multiplied in $O(N)$ time.

Many of the standard graph problems have been studied from this point of view and parallel algorithms are available for various computer architectures [83-89]. These include algorithms for finding spanning trees, connected components, and graph traversals. However, the parallel algorithms are in general not simply direct transfers of the serial-type algorithms to parallel form, for this is not usually possible. Devising a parallel algorithm often forces a completely new approach to the problem in question and may require new ways in which to represent the data so as to be able to exploit the parallel structure of the machine. Current research in this area is very intensive.

It is natural to expect parallel processing to be of considerable help in reliability computations. Hopefully we could solve many of the NP-hard problems referred to earlier in polynomial time, and for many of them this is in fact the case, but only at a high price. For example, consider the Hamiltonian tour problem referred to in the previous section. With sufficiently many processing elements, all we need to do is just dedicate each processor to the task of generating and testing its own candidate solution. Since an NP problem, by its very definition, admits to a polynomial-time algorithm for checking the validity of a solution, if the problem has any solution at all then one of the processors is guaranteed to find it, and in polynomial time. In this sense, then, any NP problem can be solved in polynomial-time. (We ignore any difficulties that might arise

when processors need to communicate or synchronize with each other, although this is a very important consideration in the design of parallel algorithms.)

Of course we now have to devote *an exponentially large number of processors* to solving the problem. As this is completely out of the question, we must settle for something less than a true polynomial-time algorithm. The minimum we want is to use parallel processing to obtain algorithms that either

(i) process our current networks *much* more rapidly than the available serial algorithms

or (ii) allow us to deal with *much* larger networks in the same time we now devote to serial calculations.

The extent to which these goals are realizeable is still being investigated. In addition to the problem of obtaining a polynomial-time algorithm, questions about algorithms using only a polynomial amount of storage are also of great interest. Dotson's method, (see below), for example, requires an amount of temporary storage which grows exponentially with the size of the network.

6. CALCULATION OF RELIABILITY

As noted above, the problem of calculating 2-terminal *st*-reliability is, in the worst case, $\mathcal{NP}$-hard. Therefore, all general reliability calculations should be expected to take an amount of time which grows exponentially with the size of the input. Nevertheless, some algorithmic approaches to the problem are much better than others. Some of the more recent algorithms require much less time to complete the computation than was required by earlier methods. This allows the exact reliability to be determined for much larger networks than were possible previously. In addition, the wide availability of inexpensive hardware with appreciable amounts of high-speed storage permit many of these calculations to be done relatively quickly in a desktop environment using personal computers.

In this section we discuss a number of the more recent approaches to performing these calculations. However, for very large networks no currently existing computer machinery is even remotely capable of computing their reliability. For such problems one must resort to approximation methods to obtain upper and lower bounds on $\mathrm{Rel}(G; s, t)$, ideally in polynomial time. A number of such methods are available but they tend to be somewhat heuristic in the sense that the error is not necessarily known, although some of these do maintain current error estimates. Also, it is not clear what type of networks these algorithms are best suited for. While a particular algorithm may appear to do very well for one particular type of network, no one can claim to be uniformly better than another. Direct comparisons between them which remain valid over all allowable inputs are not available.

Finally, if the available bounds on network reliability are not sufficient, one can resort to Monte Carlo simulation methods. If properly implemented these are capable of maintaining upper and lower bounds, but convergence is generally very slow.

6.1 STATE SPACE ENUMERATION

The most direct method of evaluating network reliability, or that of any probabilistic Boolean system, is to enumerate *every* possible state of the network. For our communications networks there are a total of 2^q such possibilities (2^{p+q} if node failures are also considered), since each edge can independently be either up or down. This decomposes the success or failure event into its most elementary subsets. It is

very easy to implement this decomposition and to accumulate the success probabilities. However, because of the large number of system states of the network, it is not practical to use such a method to carry out a general reliability calculation to its conclusion.

Rather, one can generate the most probable system states first and accumulate the results until most of the state space has been processed [90-91]. For example, suppose that the k most probable of the 2^q elementary events, say $a_1, a_2, \ldots, a_k$, have been determined. If we then define

$$\epsilon = \epsilon(k) = 1 - \sum_{a_i \in S} P(a_i), \tag{6.1}$$

and

$$\gamma = \sum_{a_i \in S} P(a_i), \tag{6.2}$$

where S is the success event (there exists an operating path from s to t), then $\mathrm{Rel}(G; s, t) = P(S)$ and

$$\gamma \leq P(S) \leq \gamma + \epsilon. \tag{6.3}$$

This method would be useful only if $\epsilon = \epsilon(k) \rightarrow 0$ rapidly as $k \rightarrow \infty$, which requires that all of the network components have a very small probability of failure. Thus it is practical for networks with very reliable components. It can also be used to estimate other network functions that would not be readily available should a much coarser partition of the state space be employed.

Thus if Q is some quantity defined over the network and having value $Q(a)$ when the network is in state a, then

$$\sum_{a_i \in S} P(a_i)\, Q(a_i) \tag{6.4}$$

is an estimate of the average value of Q over the success set. Similarly, for any constant Q_0,

$$\left(\sum_{\substack{Q(a_i) > Q_0 \\ a_i \in S}} P(a_i) \right) \Big/ \left(\sum_{a_i \in S} P(a_i) \right) \tag{6.5}$$

approximates ths probability that Q exceeds Q_0, given that the success event occurred. If we had used a partition of S into non-elementary subsets, the quantity Q might vary significantly over such subsets and the simple estimates above would not be available.

With a little more work it is possible to maintain upper and lower bounds on such estimates, including st-reliability. As such, these algorithms can monitor the

convergence of the reliability estimates, but the rate of convergence will be unacceptably slow for all but extremely reliable systems.

6.2 INCLUSION-EXCLUSION METHODS

The term inclusion-exclusion refers to a class of methods involving a decomposition of the success or failure events as the union of more elementary success and failure events, but much coarser than an enumeration of the whole state space [92-100]. As an example, consider the success event S, which occurs if there is at least one operating path from s to t. We can imagine that the set of such paths have been enumerated and consider the event S_i that all of the edges in the i-th path are operating, so S is the union of the events S_i, say $i = 1, \ldots, M$. The *inclusion-exclusion principle* is simply the formula for the probability of an arbitrary finite union of sets, we have

$$S = \bigcup_{i=1}^{M} S_i, \tag{6.6}$$

for which the probability of success is

$$\begin{aligned} \text{Rel}\,(G;s,t) = P(S) &= P\Big(\bigcup_{i=1}^{M} S_i\Big) \qquad (6.7)\\ &= \sum_i P(S_i) - \sum_{i<j} P(S_i \cap S_j) \\ &\quad + \sum_{i<j<k} P(S_i \cap S_j \cap S_k) - \ldots + (-1)^M P(S_1 \cap \ldots \cap S_M). \end{aligned}$$

The failure set F is handled in a similar way, now using a cutset enumeration. An *st-cutset* of G is a set $E_0 \subset E(G)$ for which $G - E_0$ is disconnected with s and t belonging to different components of $G - E_0$. It is said to be *minimal* if no proper subset has the same properties. Suppose that the minimal *st*-cutsets have been enumerated and let F_i be the event that all of the edges in the i-th such cutset have failed, say $1 \le i \le N$. Then F is the union of the sets $F_1, \ldots, F_n$ and the inclusion-exclusion principle applies again:

$$\begin{aligned} \text{Rel}(G;s,t) &= 1 - P(F) = 1 - P\Big(\bigcup_{i=1}^{N} F_i \Big) \\ &= 1 - \sum_i P(F_i) + \sum_{i<j} P(F_i \cap F_j) \\ &\quad - \ldots + (-1)^{N+1} P(F_1 \cap \ldots \cap F_N). \end{aligned} \tag{6.8}$$

If only a partial enumeration of the minimal *st*-paths and *st*-cutsets were available, say $S_1, \ldots, S_m$ and $F_1, \ldots, F_n$, where $m \leq M$ and $n \leq N$, then we can obtain upper and lower bounds by

$$P\Big(\bigcup_{i=1}^{m} S_i \Big) \leq \text{Rel}\,(G;s,t) \leq 1 - P\Big(\bigcup_{i=1}^{n} F_i \Big), \tag{6.9}$$

and use the inclusion-exclusion principle to evaluate these bounds.

While there are a number of techniques to determine the minimal *st*-paths and *st*-cutsets [97-100], this method as described so far has some severe drawbacks. First, the number of these events, M and N respectively, grow exponentially with the size of the network. Even worse, the number of terms generated by the application of the inclusion-exclusion principle is itself exponential again as a function of these quantities. Also, the various intersections of the original success or failure events frequently overlap or even coincide. This leads to a large amount of unnecessary calculation of the probabilities of these intersection events, many of which cancel with one another. Thus a direct application of these methods is extremely inefficient and should not be used for anything but very small networks.

Significant improvements can be made in this approach if one avoids the repeated calculation of canceling terms in these expansions [95, 96]. This is possible because the success probability

$$P(S) = P\Big(\bigcup_{i=1}^{M} S_i \Big) \tag{6.10}$$

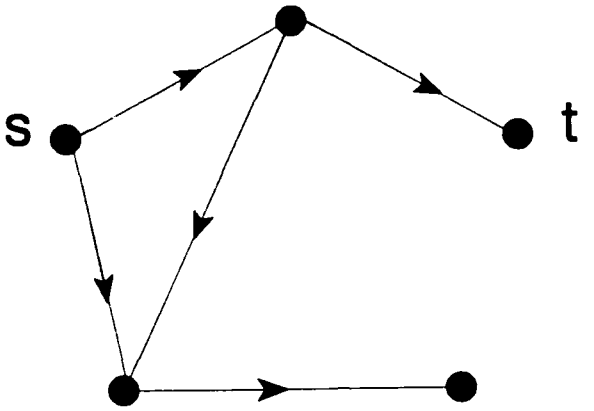
can be expressed by means of non-cancelling terms which can be put into a one-one correspondence with certain subgraphs of G. We say that a subgraph $G_0 \subset G$ is an *st-subgraph* if every edge of G_0 lies in some path of G_0 which connects the source to the sink. The non-cancelling terms in $P(S)$ will correspond to the *acyclic st*-subgraphs.

those which have no directed cycles, while terms which cancel identically result from those which admit cycles. Figure 6.1 shows the various subgraphs in a five-node network.

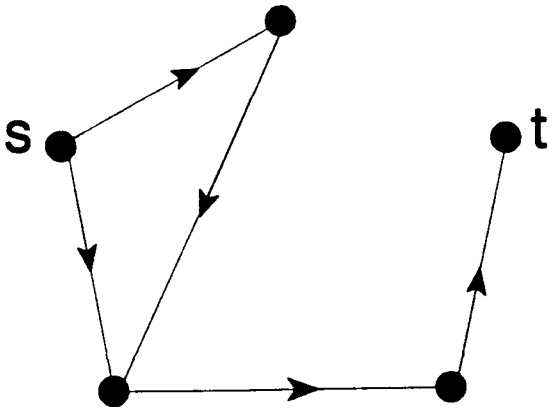
Hence in calculating $P(S)$ it is only necessary to compute the probabilities $P(G_a)$ for the acyclic *st*-subgraphs G_a of G, and there are far fewer of these than the number of terms in the inclusion-exclusion expansion formula (although the number of these subgraphs still grows exponentially with the size of the network). The acyclic *st*-subgraphs can be systematically constructed by a succession of edge replacements, and the *st*-reliability formula so generated is in a *factored* form which, if multiplied out, is equivalent to the sum of the non-cancelling monomial terms given by the inclusion-exclusion form. However, the terms that result here are *not* all positive, so terminating the process prematurely does not necessarily give a lower bound on the reliability.

These methods can also be applied to other measures of network reliability (e.g. K-terminal, all-terminal). A somewhat simpler procedure for the all-terminal reliability Rel(G) can be based on the spanning trees of G instead of *st*-subgraphs. This will give lower bounds on the reliability, but applies only to undirected networks [93]. They offer a tremendous improvement over the direct application of the inclusion-exclusion principle and make possible exact reliability computation for networks that were too large to be amenable to this type of analysis by previous algorithms. Nevertheless, there are more recent methods which perform even more efficiently and which give upper and/or lower bounds on the *st*-reliability when terminated before completion of the calculations.

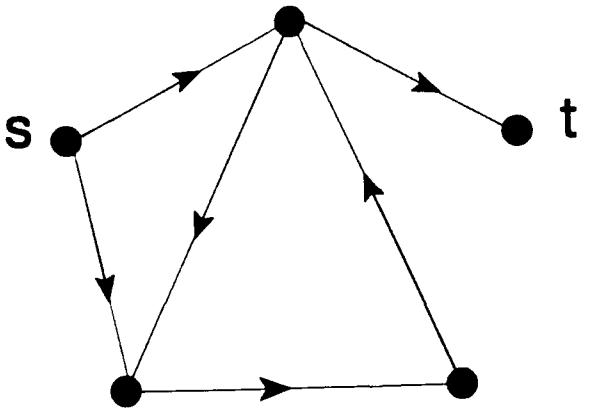
It is interesting to note [102] that if all of the minimal *st*-cutsets of G have been enumerated, say $\chi_1, \ldots, \chi_N$, then there is an algorithm for the *st*-reliability that has polynomial complexity in N. Specifically, its runtime growth is $O\left((p+q)\ N^2\right)$. While it is true that N may grow exponentially with p and q, for some networks it grows more slowly than this worst-case behavior would indicate. In general, moreover, N will increase less rapidly with the network size than the totals for other quantities used in reliability calculations, such as the number of spanning trees. Surprisingly, however, the corresponding result for paths is false (unless *every* NP-complete problem has a polynomial-time algorithm). That is, if all of the *st*-paths of G have been collected, say $\pi_1, \ldots, \pi_M$, there is no *st*-reliability algorithm having polynomial-time complexity as a function of M. In fact, this is true even for a class of graphs for which M grows



(a) Non st-subgraph
(Not every edge is in an st-path)



(b) Acyclic st-subgraph
(Every edge is in an st-path)



(c) Cyclic st-subgraph
(Every edge is in an st-path)

FIGURE 6.1 SUBGRAPHS OF K_5

only polynomially as a function of p and q, so the exponential growth in the worst-case complexity of st-reliability calculations is not due merely to the fact that the number of st-paths may be large.

6.3 DISJOINT SUMS

An alternative to using the general formula for the probability of an arbitrary union of sets, as in the inclusion-exclusion principle, is to express the success or failure event as a union of disjoint sets [101-104]. For such disjoint unions the individual probabilities simply add up directly with no cancellation. For an example of how this can be done, consider the success event

$$S = \bigcup_{i=1}^{M} S_i \tag{6.11}$$

as described above. We can define an associated sequence of success events as

$$S_1' = S_1 \tag{6.12a}$$

$$S_2' = S_2 - S_1 = S_2 \cap \overline{S}_1 \ . \tag{6.12b}$$

$$\begin{aligned} S_k' &= S_k - (S_1 \cup S_2 \cup \ldots \cup S_{k-1}) \\ &= S_k \cap \overline{S}_1 \cap \overline{S}_2 \cap \ldots \cap \overline{S}_{k-1}, \ k = 2, \ldots, M. \end{aligned} \tag{6.12c}$$

(Here $\overline{A}$ denotes the set complement of A in the state space.) It is clear that these events are mutually disjoint and that their union is S, whence

$$\mathrm{Rel}(G; s, t) = P(S) = \sum_{i=1}^{M} P(S_i') \ . \tag{6.13}$$

Obviously a similiar procedure applies to the failure event, and partial results can be obtained by truncating the process at any stage to yield upper and lower bounds on the reliability. There are a large number of variations on this theme, for there is no reason to enumerate all the success events S_i first and then construct the disjoint events S_i'. It would obviously be preferable to construct the disjoint partition of S into $S_1', \ldots, S_n'$ directly by first generating a success event S_1', then a second such set S_2' disjoint from the first, and continuing until the full success event S has been exhausted.

This is frequently done using Boolean algebra constructions and inversion (set complementation) techniques to obtain disjoint Boolean forms [101, 103, 114]. Here

the Boolean variables would be X_i, $i = 1,\ldots,q$ with X_i true if edge number i is operating, and false ($\overline{X}_i$ true) otherwise. General methods of Boolean algebra can be applied in this setting so as to attempt to minimize the number of terms involved in representing the Boolean function which represents a successful *st*-connection. That is, an economical representation of the success function would mean an efficient partition of the success event into disjoint subsets, hence a relatively small number of terms in the expression for $\mathrm{Rel}(G; s, t)$. While research is still continuing in this area [109-113], one should note that the problems in Boolean algebra associated with this approach are themselves NP-hard.

As an example, consider that probability of the failure event F, which is initially expressed as the (non-disjoint) union of the minimal *st*-cutsets. The disjoint sums are built up inductively, with each cutset contributing mutually disjoint terms which are themselves disjoint with respect to all the cutsets previously processed. Earlier algorithms of this type involved complementing individual Boolean variables one at a time, but substantially fewer disjoint terms may result if complemented products are used. A simple illustration of this is given by the two terms

$$S_1 = X_1X_2X_3 \quad \text{and} \quad S_2 = X_4X_5, \tag{6.14}$$

whose sum is to be expressed as a disjoint sum. Single term complementation of the product S gives

$$\overline{S}_1 = \overline{X_1X_2X_3} = \overline{X}_1 + X_1\overline{X}_2 + X_1X_2\overline{X_3}, \tag{6.15a}$$

thereby resulting in a four-term disjoint expansion for $S = S_1 + S_2$, namely

$$S = X_1X_2X_3 \;+\; \overline{X}_1X_4X_5 \;+\; X_1\overline{X}_2X_4X_5 \;+\; X_1X_2\overline{X}_3X_4X_5, \tag{6.15b}$$

and its corresponding reliability expression

$$P(S) = p_1p_2p_3 + (1 - p_1)p_4p_5 + p_1(1 - p_2)p_4p_5 + p_1p_2\,(1 - p_3)p_4p_5, \tag{6.15c}$$

where p_i is the probability that the variable X_i is true. Using the direct complementation of the product, however, gives a two-term disjoint sum

$$S = X_1X_2X_3 \;+\; \overline{(X_1X_2X_3)}\,X_4X_5, \tag{6.16a}$$

and its corresponding reliability

$$P(S) = p_1p_2p_3 + (1 - p_1p_2p_3)p_4p_5. \tag{6.16b}$$

Of course, these two expressions give exactly the same reliability. This simple difference can result in a large decrease in the number of terms needed in order to

express the failure event F as a disjoint sum. While the number of terms still exhibits exponential growth, the rate of growth is much less [107].

6.4 DOTSON'S METHOD

Dotson's Method [115-119] is an interesting variation on the disjoint sum approach just described in that it simultaneously generates both success and failure events as it proceeds. As these events are mutually disjoint, the success and failure probabilities can be accumulated by simple additions as the algorithm is running, so there are always available both an upper and a lower bound on the *st*-reliability. Thus the error in these bounds can be continually monitored and the algorithm can be terminated at any time, such as whenever pre-assigned convergence criteria have been satisfied. If the algorithm is run to termination then a complete description of both the success and failure events has been computed.

This method seems to have been somewhat unjustly neglected since its original publication [116], so we will give an example of applying it to a simple network later. We have recently made extensive use of it for *st*-reliability calculations, as it has a number of advantages over some of the procedures mentioned above. The method begins by finding some initial path, say π, from the source to the sink. Such a path, which must exist if the sink is reachable from the source is then the first element in a collection of success events. If the length of π is k, then each of the k edges of π are successively complemented (disabled), one at a time, and these complemented events are stored in some temporary working array, say W.

In the general stage of the algorithm we examine the events $w \in W$ one at a time and for each such event we search for an *st*-path in the graph $G_w = G - E_w$, where E_w is the set of edges in the event w under consideration which have been disabled. If this is not found then w is added to the (initially empty) collection of failure events. Otherwise a new *st*-path is found, say π'. It is added to the collection of success events and its complementary events stored in a new temporary working array W'. This procedure continues until all $w \in W$ have been so processed, at which time the whole routine is repeated again with the newly constructed working array W' in place of W. The main point of the Dotson method is the use of the complementary events to generate subsequent success and failure events, but its effectiveness depends on the particular path-finding method employed. In particular, it is important to generate the most probable success events early in the process, and efficient path-finding algorithms

for this purpose are described in [117] and [118].

If run to completion the algorithm will terminate with $W' = \emptyset$, no further events in the temporary array. In this case the success and failure events, S and F, are completely exhausted by the success and failure collections which have been accumulated during the above procedure. The *st*-reliability can then be computed from either of these collections as

$$\text{Rel}\ (G; s, t) = P(S) = 1 - P(F)\ , \tag{6.17}$$

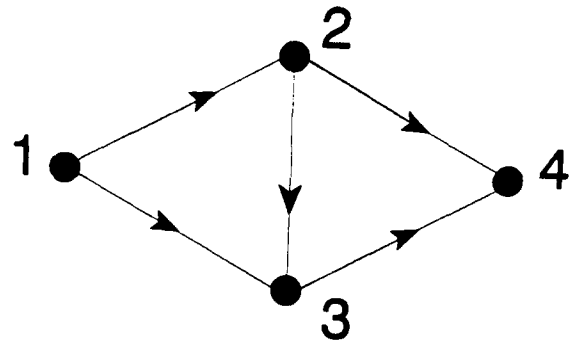
or these probabilities could have been continually updated as the algorithm proceeded to monitor the error. The relative efficiency of Dotson's method is that the success/failure partitioning of S and F constructed thereby are rather coarse. For example, if there is an edge in $E(G)$ which connects s and t, then this edge is the first success event π found and it is thereafter never considered again, immediately reducing the size of the state space by a factor of 2.

The main drawback to Dotson's method is that the intermediate storage required for the temporary arrays may be very large. (Of course, the number of success and failure events found by the algorithm will also be large, but these need not be stored if the probability updates are done right away.) Whether this is true or not depends on the order in which the events in the working array are processed. If this array is implemented as a simple stack (LIFO), then the number of temporary events that must be stored does not grow exponentially with the network size and there is no problem with intermediate storage.

However, Dotson's method should find the shorter *st*-paths earlier in the search, and since these are the more probable it is desirable to process them early in the algorithm [99]. This requires operating the temporary storage array as a *queue* (FIFO), and this will result in large storage arrays whose size will grow exponentially with the size of the network. Moreover, if the algorithm is to be terminated early then FIFO operation *must* be employed. (Stack operation would leave highly likely events still unprocessed.)

Figure 6.2 gives an example of Dotson's method applied to a simple 4-node network with all edges given a probability of 0.8 of being operational. There are four success events and five failure events and the nine corresponding network configurations are shown in Figure 6.3. Note that none of the nine subgraphs shown admit cycles. Also, each of the four success events gives a subgraph of the network in which t is reachable from s, but not all of these are *st*-subgraphs. In general, Dotson's method generates

Number of Nodes: 4
Number of Links: 5
Source Node: 1
Terminal Node: 4
Max Hop Distance: 3



Edge List:

1	2
1	3
2	3
2	4
3	4

Output of Dotson Algorithm - Summary

>>> Next working queue size: 1 <<<
>>> Next working queue size: 2 <<<
>>> Next working queue size: 4 <<<
>>> Next working queue size: 2 <<<

Number of Successes: 4
Number of Failures: 5
Number of Events: 9
Link Probability: 0.80
Lower Bound: 0.8908800
Upper Bound: 0.8908800
st-Reliability 0.8908800

Output of Dotson Algorithm - Success/Failure Events

+ 21121
+ 02112
+ 22102
– 00111
– 02110
+ 20202
– 22100
– 20001
– 20200

Shows the edge configuration of the accumulated events
Success event positive (+), failure event negative (-)
Working = '2', not working = '0', doesn't matter = '1'
Edges are enumerated as ordered in the input edge list

FIGURE 6.2 EXAMPLE OF DOTSON'S METHOD

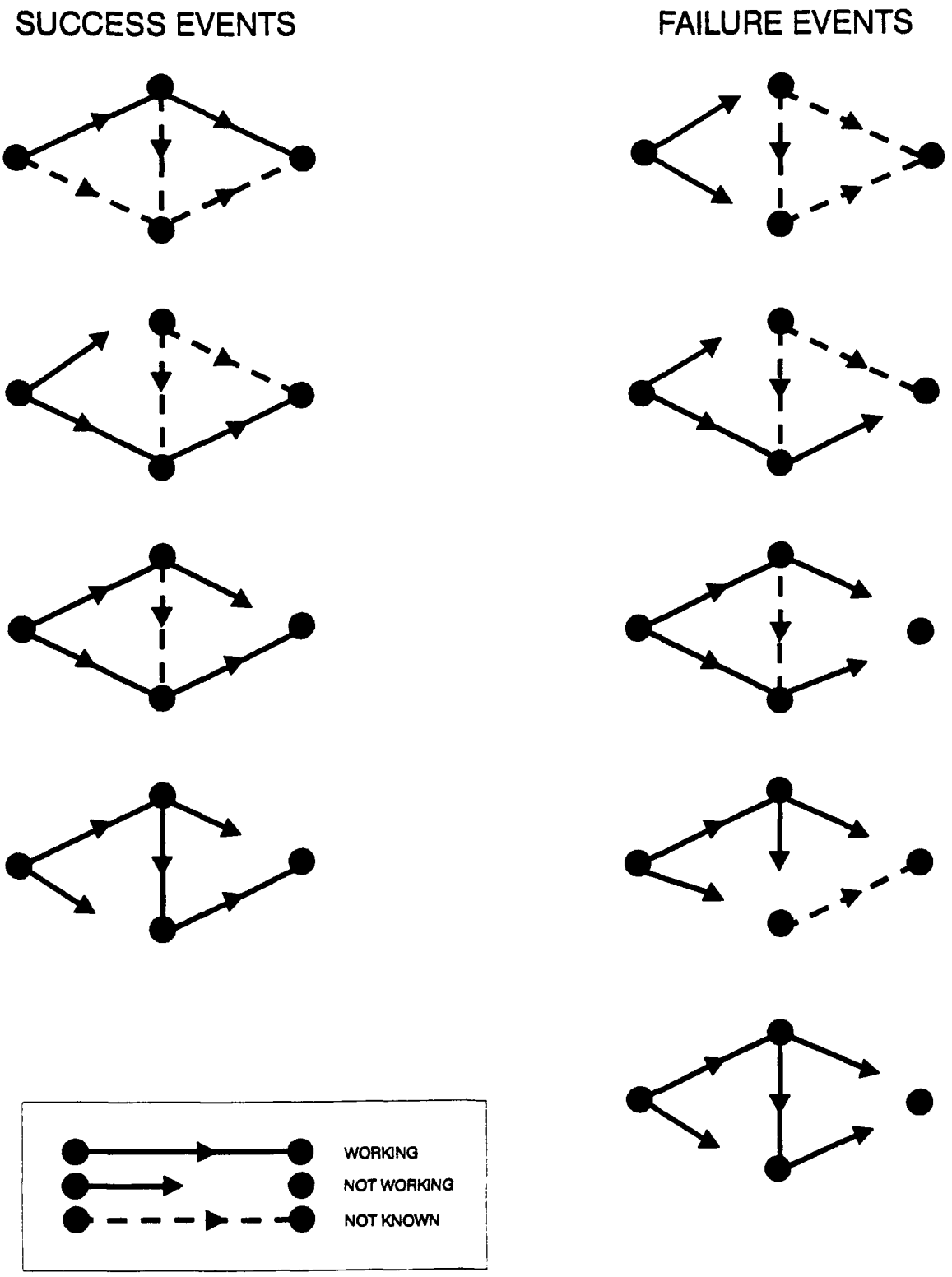


FIGURE 6.3 EXAMPLE OF DOTSON'S METHOD (CONT'D)

fewer success events than there are st-subgraphs, as used in the inclusion-exclusion methods.

The method has some advantages. For one, it is easy to restrict the hop length used in the path search routines [117, 118]. Thus for any integer n, $1 \leq n \leq q$, we can use the method to estimate the *n-hop reliability*, $\mathrm{Rel}_n(G; s, t)$, the probability that the nodes s and t are connected in G by a path of length at most n. This is useful if for some reason there is a limit on the number of relays which can be used in the node-to-node transmission of a message.

Also, it is easy to account for possible node failures in this algorithm, although doing this directly is not a good idea. (Adding node failures to the original network model increases the size of the state space by a factor of 2^p.) Much more efficient is the *equivalent links* method [117,118], whereby one runs a modified version of the Dotson method for a network initially modelling only edge failures, but instead of accumulating the success/failure probabilities we will save the success/failure events to some permanent storage file. This file can then be fitted to *any* edge probability function $P: E(G) \rightarrow \mathbb{R}$ to compute or bound the st-reliability. Moreover, if we are also given a node probability function $Q: V(G) \rightarrow \mathbb{R}$, then it is also possible to *back-fit* the edge events generated by the Dotson algorithm to account for *both* node failures *and* edge failures. This can be done at any time, even if there had originally been no intention of modelling node failures, because of the concept of an "equivalent link" that combines edge failures and node failures.

A brief description of the equivalent links method follows. Assuming (temporarily) that the source node s is completely reliable ($Q(s) = 1$), we consider that a successful communication occurs across a link exactly when both the edge in question *and* its terminal (receiving) vertex are operational. This is equivalent to replacing the probability $P(e)$ of edge $e = uv \in E(G)$ by the product $Q(v)P(e)$. Now, however, two links are independent only if they have distinct terminal vertices. Therefore, if S is any success event generated by Dotson's method, then

$$P(S) = Q(s) \prod_{v \in V(G)} P(S(v)), \tag{6.18}$$

where $S(v)$ is the restriction of the event S to those links which terminate at vertex v.

Given that node v is operational, the links in $S(v)$ are operational exactly when the corresponding edges in the original network are operational, and this observation allows us to express the probability of the event $S(v)$ in terms of the original node and edge

probabilities. Let $S_+(v)$, and $S_-(v)$, be the set of links in $S(v)$ which are specified by S to be operational, and to have failed, respectively. Let $M = |S_+(v)|$ and $N = |S_-(v)|$. Then the probability of the restricted event $S(v)$ is given by the following *back-fitting* forumulas:

$$P\big(S(v)\big) = \begin{cases} 1 & M = 0, N = 0 \\ Q(v) \prod\limits_{e \in S_+(v)} P(e) & M > 0, N = 0 \\ 1 - Q(v) + Q(v) \prod\limits_{e \in S_-(v)} [1 - P(e)] & M = 0, N > 0 \\ Q(v) \prod\limits_{e \in S_+(v)} P(e) \prod\limits_{e \in S_-(v)} [1-P(e)] & M > 0, N > 0 \end{cases} \tag{6.19}$$

Of course if the nodes are completely reliable $\big(Q(v) = 1$ for all $v \in V(G)\big)$ this gives the usual probability of the success event,

$$P(S) = \prod_{e \in S_+} P(e) \prod_{e \in S_-} [1-P(e)], \tag{6.20}$$

where S_+ (resp S_-) are those edges in S which are required to be operational (resp. to have failed).

Similiar back-fitting formulas apply to the failure events generated by the Dotson method, except that at the end of the calculations one should subtract the accumulated failure probability from $Q(s)$, instead of from 1.0, to obtain the upper bound on the *st*-reliability.

The back-fitting formulas (6.19) from the equivalent links method might also be applied to other *st*-reliability algorithms which compute a symbolic, rather than a numeric, representation of the reliability [111,113]. That is, provided sufficient information is available in the description of the edge events, it would be possible to incorporate node failures into the result. With equivalent links method, in fact, a file of edge events generated using Dotson's method can even be used to calculate *st*-reliability for networks with perfectly reliable edges and *only* unreliable nodes.

6.5 FACTORING METHODS

Edge factoring [120-126] is a very attractive method for reliability calculations of moderate size (20-25 vertices). It is based on the fact that for a given undirected

$$R(G) \;=\; P(e)\, R(G/e) \;+\; [1 - P(e)]\, R(G - e). \tag{6.21}$$

Here $R(G)$ is any of the reliability measures defined above, and G/e denotes the graph obtained by *contraction on* e. That is, we remove edge $e = uv$ from the network and identify the vertices u and v which were joined by e. In electrical engineering terms this is best thought of as replacing the edge e by a *short circuit.* Analogously, $G - e$ is obtained by the replacement of e by an *open circuit.* (See Figure 6.4). Thus the reliability problem is reduced to computing the reliability of two *smaller* networks.

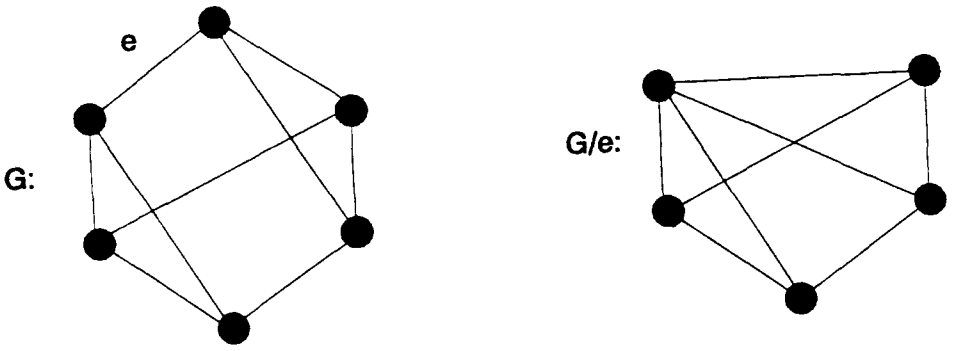


FIGURE 6.4 CONTRACTION ON EDGE e

We will consider only the 2-terminal reliability $R(G) = \mathrm{Rel}(G;s,t)$ here. The edge factorization formula in this case will also apply to directed networks if the method is restricted to suitable edges. One should factor only on an edge which either goes out of the source or goes into the sink, because the edges in the opposite directions are not relevant to the st-reliability [122]. These opposite edges can all be initially deleted from the graph with no effect on $R(G)$. Indeed, this should *always* be done, not only to the original graph G, but *also* to G/e and $G - e$.

The factorization method is intrinsically recursive in that the process should be repeated so that G/e and $G - e$ are each factored on one of their edges. Since each step reduces the size of the graphs in question by one edge (and one node for G/e), the procedure will eventually result in trivial networks. However, in this form the use of factorization is not practical since it would result in a binary tree of reduced graphs of

level q, meaning that $2^q - 1$ such graphs would have to be processed before the recursion was complete. (Some branches may terminate early if the source and the sink become disconnected as a result of the edge deletions, but this will not usually occur.)

What makes this method feasible is that it can be combined with *edge reduction* methods to greatly reduce the size of the recursion tree. For example, contraction on an edge frequently results in *parallel* edges (two edges incident on the same two nodes) which can be combined into a single edge. Similarly, parallel reduction can result in *serial* edges for which another reduction is possible. Because of the assumed independence of the edge failures, the new edge probabilities can easily be determined so that the reliability of the reduced network is the same as that of the original. These edge reductions are shown in Figure 6.5.

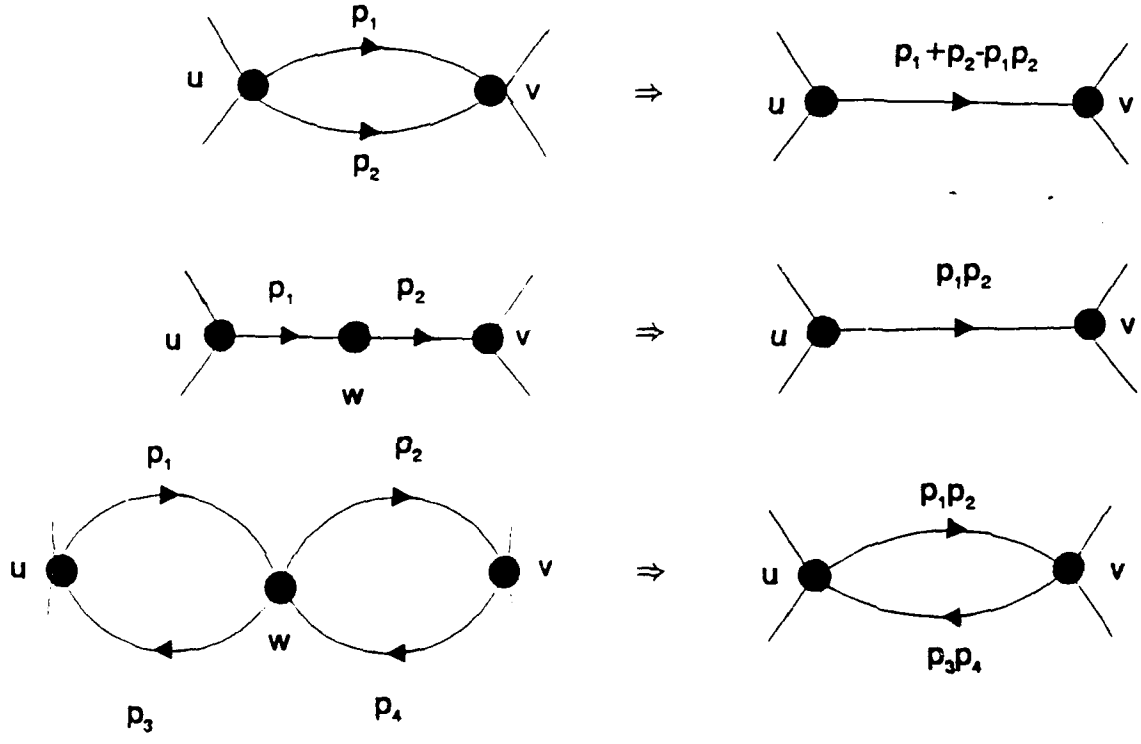


FIGURE 6.5 PARALLEL AND SERIES EDGE REDUCTION ($w \neq s,t$)

Additional reductions are often possible, for if $v \neq s$ is such that indeg$(v) = 0$ (v is a *false start* node) or if $v \neq t$ has outdeg$(v) = 0$ (v is a *dead end* node) then v and its incident edges can be deleted from the network. Similarly, any pendant node other than the sink and the source may be removed. None of these deletions has any effect on the network reliability (see Figure 6.6).

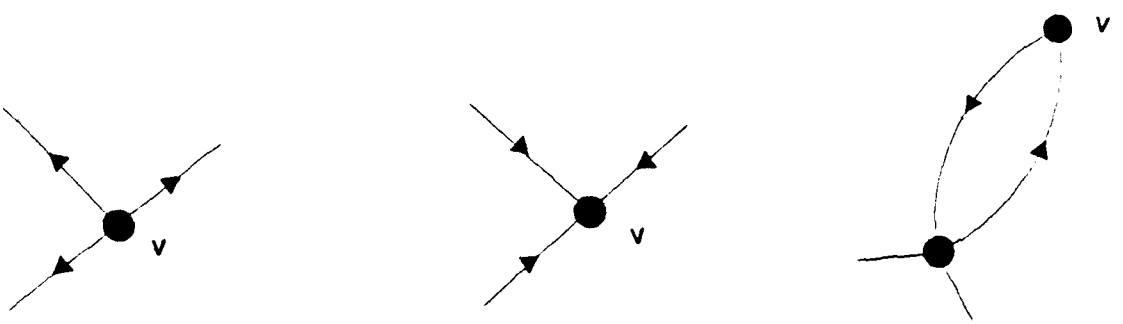


FIGURE 6.6 FALSE START, DEAD END, PENDANT NODES ($v \neq s,t$)

Finally, for directed networks an important type of reduction can be used on vertices which have only one incoming or outgoing edge [123]. Here, if the incoming edge has an outgoing edge in the opposite direction, then that outgoing edge may be removed if the node in question is not the source node. The other case is similar, and examples are shown in Figure 6.7.

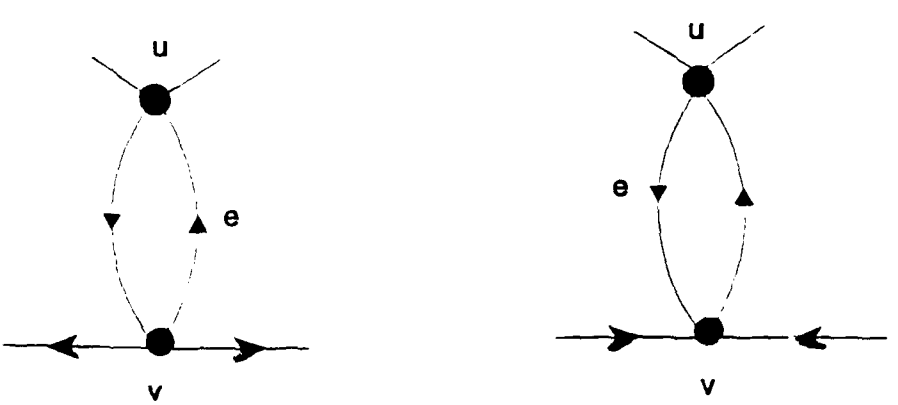


FIGURE 6.7 REMOVE EDGE e IF $v \neq s,t$

The implementation of an efficient factorization algorithm is then to repeatedly apply to G all the allowable edge reductions until no further simplifications are possible. One then applies the factorization formula for some edge $e \in E(G)$, say one going out of the source node, and repeats this process to G/e and $G - e$ before they are factored on one of their edges. The result is often a very rapid loss of edges and a drastically

one of their edges. The result is often a very rapid loss of edges and a drastically pruned binary tree for the recursive function calls. For example, if either G/e or $G-e$ is ever a series-parallel graph, then the edge reduction will reduce it to a trivial graph and factorization is never applied again. While it is difficult to predict in advance how great a reduction in the recursion tree actually takes place, these methods have proved to be quite competitive with those already discussed.

One great advantage of this method is that very little computer memory is required. Even when written in a recursive form in a language such as Pascal which supports recursive function calls and uses available stack space for the storage of local variables, memory availability should not be a problem. Roughly speaking, the worst-case memory requirement is $O(q^2)$, but this will be completely dominated by the exponential growth in the runtime.

A disadvantage is that a change in the edge probabilities require that the entire algorithm be repeated. With edges being merged with one another or factored out of the calculations, the edge probabilities are being combined in an analogous way and the algorithm does not keep track of just how this was done. While it might be possible to save this information so as to reprocess the graph with a new set of edge probabilities, doing so would require a large amount of storage to record all the branching in the recursion tree, losing the low memory advantage.

Finally, the factorization formula (6.21) can be modified to apply to the situation where both nodes and edges can fail independently, without greatly increasing the complexity of the algorithm [117, 125]. The modifications are analogous to the equivalent link method discussed in the previous section on the Dotson method. We shall give a brief description of how this is done, and as before we let $Q: V(G) \rightarrow \mathbb{R}$ be the associated node reliabilities.

The initial problem here is that the contraction of G on an edge $e = uv$ is to result from a graph in which the communication from node u to node v is possible, hence u and v must be operational and should be merged into a new node that is completely reliable. (That is, if either u or v had failed we could not have had a successful edge connection across this link.) Thus the end points u and v as well as the edge $e = uv$ itself must all be operational, and this occurs with probability

$$P'(e) = Q(u)\, P(e)\, Q(v) \tag{6.22}$$

because of the statistical independence. However, the failure of the link can be due to either the failure of the edge itself or that of either of the two nodes. While we may not

know which component has failed, we may delete the edge from the graph anyway since the failure of a node implicitly implies the failure of all edges that are incident to it. Thus the factorization formula will hold if the node reliabilities of the depleted graph $G - e$ are modified appropriately, which implies that we use

$$Q'(v) = \frac{Q(v)\,[1-Q(u)P(e)]}{1-P'(e)} = \frac{Q(v)-P'(e)}{1-P'(e)} \tag{6.23}$$

with a similar modification for the node u.

The immediate difficulty with this approach is that the node failures in the graph $G-e$ are no longer statistically independent. We could avoid this problem if we only factored on an edge $e = uv$ for which the initial node u was perfectly reliable, for if $Q(u) = 1$ then $Q'(u) = 1$ and

$$Q'(v) = \frac{Q(v)\,[1-P(e)]}{1-P(e)Q(v)} \tag{6.24}$$

still describe statistically independent failures. However, we can always assume that the source node s is perfectly reliable, so we will maintain statistical independence throughout the process if, as in the Page-Perry algorithm, we always perform the factorization on edges leaving the source node, so

$$R(G) = P'(e)R(G/e) + [1-P'(e)]R(G-e) \tag{6.25}$$

for such edges, the edge and node reliabilities in the derived graphs G/e and $G-e$ having been adjusted as described above. (The true reliability $Q(s)$ of the source node is included as a factor of the *st*-reliability at the end of the recursion.)

As with the use of the factorization theorem for edge failures only, a practical algorithm results only when it is combined with the edge reduction schemes described above. However, all of these can still be applied, with only minor modifications, to account for node failures as well. Aside from this observation, the factorization calculation then proceeds exactly as before.

This modification of the factorization theorem and the equivalent links algorithm using Dotson's method are the best available methods for computing *st*-reliability for networks in which both node and vertex failures are modelled. Each has its own advantages and disadvantages. The factorization technique is generally much faster (when combined with the appropriate edge reduction algorithms) and uses very little

intermediate storage, but any change in the edge and/or node reliabilities require that the algorithms be repeated from the beginning. The Dotson method, however, always maintains bounds on the reliability. Moreover, if a permanent record of the success/failure events is maintained then these can be fitted over and over again to any distribution of the component reliabilities without repeating the original computation. Also, while it is possible to truncate the recursion tree in the factorization method so as to obtain upper or lower bounds in a smaller time, the method does not allow a continual monitoring of its progress as it proceeds.

6.6 APPROXIMATION AND SIMULATION

Since the exact calculation of the graph reliability is NP-hard, one inquires after efficient methods for computing upper and lower bounds [127-138]. We have already referred to some of these in an earlier section when we assumed a constant edge probability. These reduced to some graph enumeration problems, although still NP-hard. There are a number of algorithms of this type. We discuss in some detail a so-called edge-packing algorithm which will apply to the case of arbitrary edge probabilities [134, 135]. As above, we consider only the 2-terminal st-reliability and use the sets of st-paths and st-cutsets. We conclude with some references to Monte Carlo applications to reliability calculations.

An edge-packing of a digraph $G = G(E,V)$ is simply a collection $\{G_1, \ldots, G_n\}$ of subgraphs of G which have no edges in common. Bounds on the st-reliability can be obtained by imposing appropriate restrictions on the type of subgraphs considered. Let us first suppose that each such subgraph is an st-path, say π_k, and define the probability of its success simply as the probability that each edge is operational. Thus, by the independence assumption on the link failure events,

$$P(\pi_k) = \prod_{e \in \pi_k} P(e). \tag{6.26}$$

Now if the nodes s and t are not connected in G (t not reachable from s) then *all* of the st-paths $\pi_1, \ldots, \pi_n$ must fail. (The converse is false, however, since there may be other st-paths in addition to the given $\pi_1, \ldots, \pi_n$.) Since these paths are assumed to be edge-disjoint,

$$1 - \mathrm{Rel}(G;s,t) \leq \prod_{k=1}^{n} [\,1 - P(\pi_k)] \tag{6.27a}$$

or

so this type of edge-packing gives a *lower* bound on the *st*-reliability. Given such a lower bound we can look to improve it by attempting to

(i) find an additional *st*-path π_{n+1} edge-disjoint from $\pi_1, \ldots, \pi_n$

(ii) replace one of these *st*-paths by another which is more reliable but still edge-disjoint from the remaining paths.

These two aims are somewhat contradictory, and it is not obvious how one should choose an edge-packing by *st*-paths so as to optimize the lower bound. One should note, however, that the maximum number of such paths is known from a variation of Menger's theorem [3]:

> The maximum number of edge-disjoint *st*-paths is equal to the minimal number of edges in an *st*-cutset.

However, an *st*-path edge-packing using this maximum number of such paths does not necessarily give a good lower bound. Given a choice between using two long paths and one short path (or vice versa), either selection might be more reliable than the other and result in a better lower bound. A recent article of Torrieri [137] presents some algorithms for generating disjoint *st*-paths. Also note that since edge-disjoint *st*-paths are necessarily also node-disjoint, it is easy to include node failures in the lower bound estimates.

To obtain an upper bound we instead make use of an edge-packing $\{\chi_1, \ldots, \chi_n\}$ consisting of disjoint *st*-cutsets [138]. If the network is operating in such a way that t is reachable from s, then *each* cutset must fail to separate t from s (but not conversely). That is, each cutset $\chi_1, \ldots, \chi_n$ must contain at least one operating link. For an *st*-cutset χ let $Q(\chi)$ denote the probability that χ fails to disconnect t from s. Since it cannot be true that every edge $e \in \chi$ has failed,

$$Q(\chi) = 1 - \prod_{e \in \chi} [\, 1 - P(e)], \tag{6.28}$$

so we obtain an *upper* bound on the *st*-reliability

$$\mathrm{Rel}(G; s,t) \;\leq\; \prod_{k=1}^{n} Q(\chi_k). \tag{6.29}$$

To improve this bound we would like to

(i) find an additional *st*-cutset χ_{n+1} edge-disjoint from $\chi_1, \ldots, \chi_n$

(ii) replace one of the *st*-cutsets by another which has a smaller value of Q but is still edge-disjoint from the others.

Again, these two goals are in opposition. Also, there is an elementary analogue of Menger's theorem that holds in this situation [3]:

> The maximum number of edge-disjoint st-cutsets is equal to dist(s,t), the length of the shortest path from s to t

As with the st-path edge-packing, it is not clear how one should go about optimizing the resulting upper bounds, although it is clear that only minimal st-cutsets should be considered as candidates for the edge-packing.

It is clear that for any network there is an optimal edge-packing of the desired type which gives that best corresponding bound on the st-reliability, but the optimum will in general depend on the edge-probabilities. No efficient algorithms are known for doing this, but it is also *not* established that the problem is NP-hard. In any case these procedures give a quick method of finding some bounds on Rel $(G;s,t)$. It is not clear how good these bounds are in general, but in test cases they are competitive with other algorithms, and they do apply to networks with arbitrary edge probabilities. The methods of Torrieri [137] and Wagner [138] can be used to generate edge-packings of the two types described here.

However, it is important to note that even the optimal bounds cannot be uniformly good over all possible network inputs, for we have the following [43,120]:

> Given a probabilistic network G, distinct nodes $s,t \in V(G)$, and a number $\epsilon > 0$, find r such that $|$ r - Rel $(G;s,t)| < \epsilon$. This problem is NP-hard, even for constant edge probabilities.

At this point, one might ask if the expected worst-case exponential growth in the calculation of network reliability is typical behavior. For example, the well-known simplex algorithm in linear programming does require an exponential runtime for some input data sets, but such examples are somewhat contrived. For real-world applications the simplex performance proves to be quite satisfactory. Unfortunately, this is definitely *not* the case for the graph reliability algorithms. Even for very simple network configurations, such as rectangular grid networks, the rapid growth of the run time is obvious. (However, the linear programming problem is now known *not* to be NP-hard. It does admit solution by another algorithm which does have polynomial complexity.)

Finally, a number of papers [139-145] have dealt with Monte Carlo methods in simulating the operation of the network so as to obtain approximate values of the network reliability. We will not go into details here, but in principle this is simple to do. Assuming statistical independence with given edge probabilities $P(e)$, $e \in E(G)$, we draw a sample of the state of the network, say $X = (X_1, \ldots, X_q)$ with X_i true or false according to whether the i-th edge is operating or not. Then either one of the two graph traversals described earlier (DFS or BFS) can quickly determine if s and t are connected in the graph given by the configuration X, so t would be reachable from s with probability $P(X)$ if this is the case. Averaging this value over an ever larger number of random samples of the state of the network gives increasingly accurate approximations to the reliability $\mathrm{Rel}(G; s, t)$, and the variance of these approximations can be estimated so as to monitor convergence as the sample size increases. Such convergence, however, will be very slow.

BLANK PAGE

7. OTHER PERFORMANCE MEASURES

The term "reliability" has generally been reserved to refer only to the probability of the network being connected with respect to some subset of edges. Other measures of network performance can be used to quantify the susceptibility of the network to component failures, but such terms as "vulnerability" or "survivability" have no generally accepted mathematical definitions. In this section we introduce several other graph invariants that have application to the general question of network vulnerability. While these may not have the same intuitive appeal as that associated with *st*-path connectivity, some are easier to calculate and are more appropriate to special types of networks than the reliability measure.

We first discuss some quantities relating to somewhat different aspects of graph connectivity as defined earlier, and in later sections consider other graph invariants which are not immediately related to connectivity. Most of these, however, are motivated by an attempt to quantify the effect that the removal of a set of nodes or edges will have on the network. For example, knowing that $\lambda(G) = k$ for a given connected graph G tells us only that there is some set $E_0 \subset E(G)$ of k edges whose removal will disconnect G. This says nothing about how badly disrupted the network might have become. Perhaps only one vertex has been isolated by these deletions, or the network could have been bisected into two pieces of nearly equal size. One would presumably prefer the first configuration so that the number of node pairs that can still communicate is as large as possible, but the value of the edge-connectivity alone has nothing to say about this.

There are a number of general schemes to quantify network survivability, most of which give the change in some specific graph invariant as a function of the size of the set of network components which were removed, which for us will generally mean edge removal. Also, we will restrict ourselves to the deterministic aspects of these quantities without reference to probabilistic questions of component failures. It should be emphasized that much of this material is of recent origin, so the extent of its application and usefulness to network analysis is not yet established. Finally, the problem of finding efficient algorithms for the calculation of the invariants in question is not at all well developed.

7.1 CONNECTIVITY FACTORS

If a graph is not connected it is normally analyzed one component at a time. Quantities of particular interest would be the number of components and the size (nodes and edges) of the largest and smallest components. Depending on the purpose of the communication network in question we might want to have either the largest or the smallest component of the graph to have as many nodes or edges possible. The worst case is obviously that of a *totally disconnected* graph—every node is isolated and there are no edges. Some measures of network vulnerability that have recently been investigated serve as indicators of how susceptible a network is to being reduced to such a configuration [146-152].

We shall describe here the *node-connectivity factor* (NCF), which represents the average number of nodes that should be removed from the network in order that the remaining subgraph be totally disconnected. This quantity is best defined recursively. First, if G is disconnected and has components $G_1, \ldots, G_m$, then the NCF is simply additive.

$$\mathrm{NCF}(G) \triangleq \sum_{i=1}^{m} \mathrm{NCF}(G_i). \tag{7.1}$$

For G a connected graph let $k(G)$ be the size of the smallest set $V_0 \subset V(G)$ for which $G - V_0$ is disconnected and let $\gamma_1, \ldots, \gamma_n$ be the collection of all such node sets V_0. Then

$$\mathrm{NCF}(G) \triangleq k(G) + \frac{1}{n} \sum_{i=1}^{n} \mathrm{NCF}(G - \gamma_i) \tag{7.2}$$

Thus the NCF of a connected graph G is defined in terms of its value on some of the connected subgraphs of G. This sets up a recursion which stops when the cutset removals have reduced G to the trivial case, a single isolated node, which has NCF equal to zero.

The recursive calculation of this quantity results in a so-called *decomposition diagram*, which is just the recursion tree of connected subgraphs formed as the minimal cutnode sets are deleted. Unfortunately, the size of this tree grows exponentially with the size of the graph and it is possible to compute the NCF only for small networks. Some research has been done to investigate the effect of terminating the recursive branching prematurely [148] so as to obtain an approximate value of NCF(G). Another method used to control the algorithm is to stop further function calls when connected subgraphs of known type are generated [149, 151]. These would be special graphs (e.g.,

stars, complete graphs, cycles) whose NCF's are already known and have been stored in some data base which is available to the algorithm. As soon as one of these graphs is encountered in the decomposition diagram, then its NCF value can be obtained directly and no further branching to disconnect this graph is necessary.

It is also possible to measure the importance of any individual node to the value of the NCF by calculating a weighted sum of the number of terms (i.e., connected subgraphs) of the decomposition diagram in which that node occurs. In this way one has some measure of the relative contributions of the various $v \in V(G)$ to the network connectivity. This information could be used to re-allocate the network links so as to equalize the various contributions of the graph vertices, thereby making the network less vulnerable to node failures (that is, increasing the value of the NCF).

A similary quantity, the *link-connectivity factor* (LCF), has also been investigated. This uses the spanning trees of the components of G and is a measure of the average contribution of the network links to maintaining a minimally connected configuration. For a connected (p,q)-graph G it is defined as

$$\mathrm{LCF}(G) \triangleq \left(\frac{p-1}{q} \right) T(G), \tag{7.3}$$

where $T(G)$ is the number of spanning trees of G. (We remark that $T(G)$ can be given as the determinant of a matrix, analogous to the adjacency matrix, constructed directly from the list of edge connections. Hence this number, although large, is *not* prohibitively expensive to compute [20].) If G is not connected but has components $G_1, \ldots, G_n$, then *its* LCF is given as [142].

$$\mathrm{LCF}(G) \triangleq \frac{1}{S} \sum_{i=1}^{n} S_i \, \mathrm{LCF}(G_i) \tag{7.4}$$

where S_i is the number of edges in a spanning tree of G_i, $i = 1, \ldots, n$ and S is the number of edges in a spanning tree for G (that is, one obtained by interconnecting the spanning trees of the G_i). It is easier to calculate than the NCF, as it requires only the total number of spanning trees of each component of G, not the actual collection of all such trees. Similarly, a method of rating the relative contributions of the individual edges to the LCF can be defined. As with the NCF, network vulnerability would presumably be decreased by revising the network links so as to spread these contributions more evenly throughout the network and increase the value of LCF(G).

Typically these heuristics are used in the problem of designing or synthesizing survivable networks or improving the survivability of a given network. That is, one has at least some control over how the network is to be configured, or some freedom in

rearranging a given configuration. The NCF and LCF values then can indicate the advantages of one particular resource allocation versus another, hopefully leading to an optimal arrangement. The result would be a network in which no one part of the network is much more vulnerable to component failure than any other [147].

7.2 NETWORK DIAMETER

Given a graph G, the *diameter* of G is the maximum hop distance supported by the edge configuration,

$$\text{Diam}(G) = \text{Max}\{\text{dist}(u, v): u, v \in V(G)\}, \tag{7.5}$$

which is to be interpreted as $+\infty$ if G is not connected. A number of recent researches [153-166] have used this as a measure of connectivity, a graph with small diameter being considered as better-connected than one with a larger diameter. This allows a more quantitive measure of connectivity. For example, whereas a graph is connected if and only if $\text{Diam}(G) < +\infty$, a separating set of edges $E_0 \subset E(G)$ is one for which $\text{Diam}(G - E_0) = +\infty$. Rather than looking for such a drastic change, we may be more careful and ask for more detail about the difference between $\text{Diam}(G)$ and $\text{Diam}(G - E_0)$ for various subsets E_0 of $E(G)$. Other authors have used the average distance between nodes instead of the diameter [155, 158, 160, 165].

An early example of this is the so-called *persistence* of a connected graph, which is defined to be the size of the smallest set $E_0 \subset E(G)$ with $\text{Diam}\ (G) < \text{Diam}(G - E_o)$. That is,

$$\text{Pers}(G) = \text{Min}\{\ |\ E_0\ |\ : E_0 \subset E(G),\ \text{Diam}(G) < \text{Diam}(G - E_0)\}. \tag{7.6}$$

A number of related quantities will be considered in this section, mainly by specifying the size of the set of nodes or edges deleted from the network. However, we shall restrict ourselves primarily to edge deletions. This is because, as already noted earlier for path connectivity, the removal of a set of nodes from a graph can give a wide variety of distinctly different results. An example of this for a connected graph is shown in Figure 7.1. In this example, apart from from the case where the graph becomes disconnected, node removal can either increase *or* decrease the diameter. This sort of behavior is not possible for edge removal, which *always* results in an increase in the diameter [163, 166]. That is, the deletion of edges can never improve the connectivity measure, and this monotone property is very advantageous.

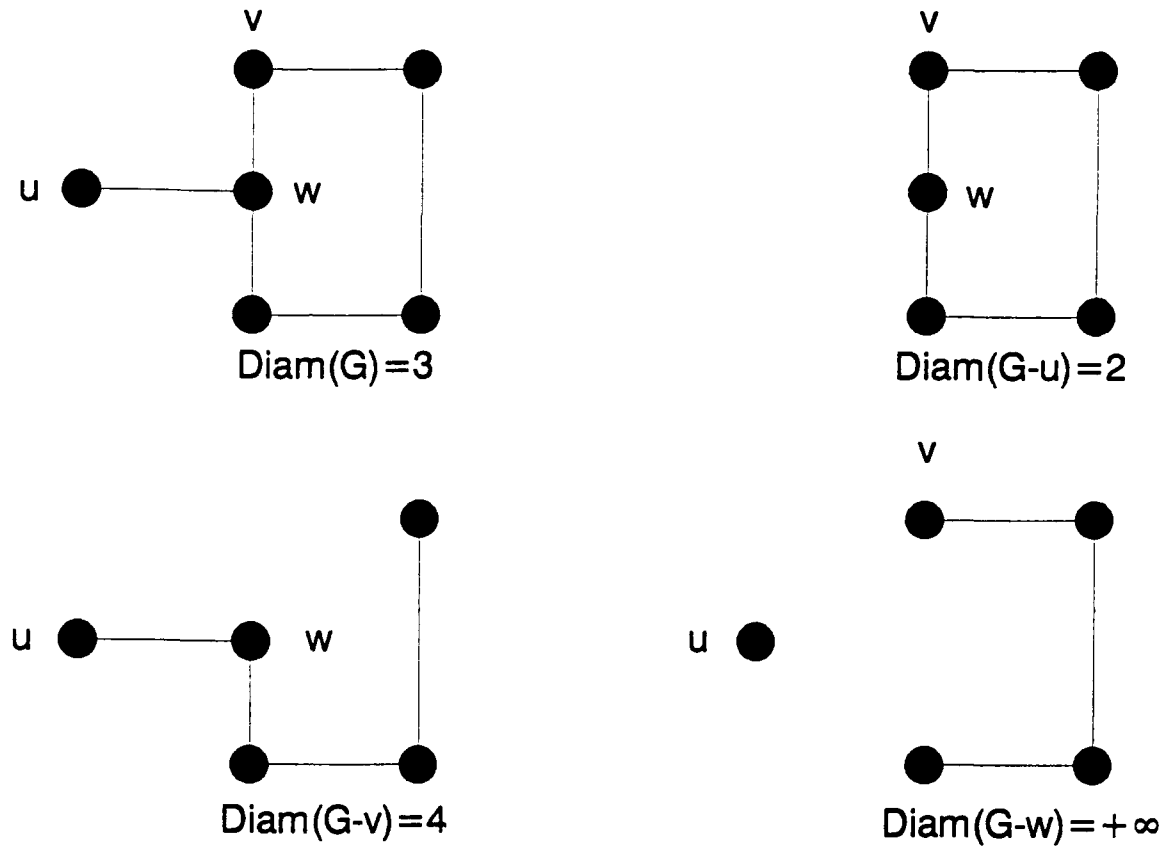


FIGURE 7.1 CHANGE IN DIAMETER VIA NODE DELETION

A more informative quantity can be defined under the general concept of *leverage*, [153] which allows a very broad method of quantifying changes in a given graph invariant with respect to the number of network components under consideration. Restricting ourselves to the diameter measure and edge deletions, we can define the *leverage* (more precisely, the *leverage sequences*) of G as

$$L^-(k) \triangleq \text{Max}\{\text{Diam}(G - E_0) - \text{Diam}(G) : E_0 \subset E(G), \mid E_0 \mid = k\}, \quad (7.7a)$$

$$\ell^-(k) \triangleq \text{Min}\{\text{Diam}(G - E_0) - \text{Diam}(G) : E_0 \subset E(G), \mid E_0 \mid = k\}. \quad (7.7b)$$

The *negative* superscript here refers to the fact that k edges of the graph have been *deleted*, but we may well want to know the effect of adjoining edges to the network. This leads to the definitions

$$L^+(k) \triangleq \text{Max}\{\text{Diam}(G) - \text{Diam}(G + E_1): \mid E_1 \mid = k\}, \quad (7.8a)$$

$$\ell^+(k) \triangleq \text{Min}\{\text{Diam}(G) - \text{Diam}(G + E_1): \mid E_1 \mid = k\}. \quad (7.8b)$$

Here the maximum and minimum are over all sets E_1 of k edges which are *complementary* to G. (That is, we consider G as a subgraph of K_p and add edges of K_p which are *not* in $E(G)$.)

The leverage sequences have a lot of information in them and subsume many more elementary graph-theoretic quantities. For example, the persistence is simply

$$\text{Pers}(G) = \text{Min}\{k \geq 1 : L^-(k) > 0\}. \tag{7.9}$$

Knowing that $k = \text{Pers}(G)$ tells us that the diameter of G is increased by the removal of some set of k edges, but it does not indicate the magnitude of this increase. Another example of this is the definition of a *critical* graph, which here means edge-critical (with respect to the diameter). For this we want $\ell^+(1) > 0$, so the diameter of G is increased if *any* single edge of G is removed.

Definitions such as leverage are sometimes referred to as *second-order* measurements [153, 154], meaning that the originally defined diameter is a *first-order* quantity and the leverage is defined in terms of changes in that term. The number of possibilities here is without limit, for in place of $\text{Diam}(G)$ we could substitute *any* other graph invariant (e.g., the number of components, minimal or maximal degree, etc). For example we would get a mixed connectivity measure if we replaced $\text{Diam}(G)$ with the vertex-connectivity $\kappa(G)$—we would be measuring the change in *vertex* connectivity as a function of the number of *edges* being added or removed.

Research into the availability and applicability of these quantities is quite recent, and most of the work to date has been devoted to finding upper bounds on the leverage and determining its extremal values over a restricted class of graphs. We cite some of these results to illustrate the behavior of these ideas.

(i) If $\lambda(G) \geq 2$ then $L^-(1) \leq \text{Diam}(G)$. Otherwise stated, $\text{Diam}(G - e) \leq 2\ \text{Diam}(G)$ for all edges $e \in E(G)$.

(ii) If $\lambda(G) = n$ with G an n-regular graph, then $L^-(n-1) > 0$.

(iii) Of all the n-regular graphs with $\lambda(G) = n$, the n-dimensional hypercube Q_n has the most resistance to diameter increases by the removal of edges. Its leverage sequence is $(0,0,\ldots,0,1,+\infty)$, where the 1 occurs in the $(n-1)$th position.

With respect to this last example, we note that the n-dimensional *hypercube* Q_n can be defined by specifying its node set to be the integers $\{0,1,\ldots,2^n - 1\}$ and connecting nodes i and j whenever the binary representations of i and j differ in exactly one binary digit. We have $\text{Diam}(Q_n) = n$ and no removal of less than $n-1$ edges will change the diameter. It is possible to increase the diameter to $n+1$ by removal of some set of $n-1$ edges, and to disconnect Q_n by removing some other set of $n+1$

edges. Thus the graph Q_n is an extremal graph for this leverage sequence in the class of all n-regular graphs with edge-connectivity equal to n.

This is the type of behavior a network should have in order to be relatively invulnerable to link failures, but the class of graphs referred to here is not appropriate for mobile communication networks. More research needs to be done in applying these concepts to communication networks, and for many of these quantities no efficient algorithms are as yet available for their calculation.

7.3 MEASURES OF VULNERABILITY

Several other quantities of relevence to the vulnerability of a network can be defined in terms of graph connectivity [167-179]. Given $t \geq 0$, a connected graph G is said to be *t-tough* if for all $V_0 \subset V(G)$ for which $G - V_0$ is disconnected we have

$$t \leq |V_0| \,/\, c(G - V_0). \tag{7.10}$$

The *toughness* $\tau(G)$ is given as [3]

$$\tau(G) \triangleq \text{Max}\{t \geq 0 : \text{G is t-tough}\}. \tag{7.11}$$

This graph invariant has been used in researches on Hamiltonian cycles and to obtain lower bounds on the *circumference* (the length of the longest cycle) of a graph. As defined above the toughness depends on the removal of a set of vertices of G, but an obvious analogue with edge deletions gives meaning to the *edge-toughness* of a graph.

Another quantity of interest refers to the quantity $m(G)$, the number of nodes in the largest component of G. Using this we can consider the so-called *integrity* [146]. For $V_0 \subset V(G)$ we define

$$\text{Int}(V_0) \triangleq | V_0 | + m(G - V_0), \tag{7.12}$$

and the integrity of the graph is

$$\text{Int}(G) \triangleq \text{Min}\{\text{Int } (V_0) \colon V_0 \subset V(G)\}. \tag{7.13}$$

(Strictly speaking we should refer to this as the *vertex-integrity* of G, with a corresponding *edge-integrity* given by deleting edges of G instead of vertices.) Minimizing $\text{Int}(V_0)$ over $V_0 \subset V(G)$ involves a trade-off between forcing $|V_0|$ to be small against making $m(G - V_0)$ small. Thus networks become less vulnerable to component failures as the integrity measure becomes larger, for if G' is a subgraph of G then it is always true that $\text{Int}(G') \leq \text{Int}(G)$.

Finally, the very definition of graph connectivity can be made relative to other graph-theoretic conditions. That is, if $\wp$ is some graph property satisfied by a connected graph G, we could require that any disconnection of G result only in connected components that also satisfied property $\wp$ [162, 164].

Measures of graph vulnerability can also be given directly in terms of the vertex-to-vertex connections specified by the edges of the graph. A set $V_0 \subset V(G)$ is said to be a *dominating set* if every $v \in V(G) - V_0$ is adjacent to some vertex $v_0 \in V_0$. A dominating set of smallest cardinality is called a *minimum dominating set* and its cardinality, denoted by $\sigma(G)$, is the *domination number* of G [4, 13, 20]. For example, if some of the nodes in a communication network had only a receiving capability (they cannot transmit, hence cannot act as relays), then the transmitting nodes should be a dominating set—the receiver nodes must have direct communication links to the transmitting vertices. We observe that if G' is a spanning subgraph of G, then $\sigma(G') = \sigma(G)$ and that consequently $\sigma(G - E_0) \geq \sigma(G)$ for every set of edges $E_0 \subset E(G)$.

The domination number itself is not a good measure of vulnerability, for both the complete graph K_p and any star have $\sigma(G) = 1$. Of more interest [171] is the *bondage number* $b(G)$, which is the smallest number of edges whose removal increases the domination number, or

$$b(G) = \text{Min}\{|\ E_0\ | : E_0 \subset E(G),\ \sigma(G) < \sigma(G - E_0)\} \tag{7.14}$$

(This could also be defined in terms of a leverage measured with respect to the domination number.) Thus the deletion of some set of $b(G)$ edges will destroy the domination properly of *every* minimum dominating set of G, requiring that at least one more transmitter would be necessary to restore full communication.

Upper bounds on the bondage number can be given in terms of several other graph invariants, including the domination number $\sigma(G)$. If we assume that G is connected and non-trivial, then the following results can be found in the literature:

(i) $b(K_p) = \lceil p/2 \rceil$

(ii) $b(G) \leq p - 1$

(iii) $b(G) \leq p + 1 - \sigma(G)$

(iv) If $\sigma(G) \geq 2$, then $b(G) \leq \Delta(G)\,[\sigma(G) - 1] + 1$

(v) $b(G) \leq \text{Min}\{\deg(u) + \deg(v) - 1:\ u, v \in V(G), \text{ adjacent}\}$

For good survivability, $b(G)$ should be large, so trees and paths are most vulnerable since their bondage number is small. In fact, for any tree T we have $b(T) \leq 2$, with

$b(T) = 1$ if any $v \in V(T)$ is adjacent to two or more pendant nodes.

There are many variations on the definition of dominating sets. For example, given $V_0 \subset V(G)$ and any integer $k \geq 1$, we could require that each $v \in V(G) - V_0$ be adjacent to at least k vertices of V_0. Alternatively, we could ask that each $v \in V(G) - V_0$ be within hop distance at most k to some $v_0 \in V_0$, and in both of these situations the case $k = 1$ is the usual concept of domination. A slightly stronger constraint is that these conditions hold for all $v \in V(G)$, not merely those in $V(G) - V_0$. Also, one may consider the analogous concept of *edge-domination*, where $E_0 \subset E(G)$ is said to dominate G if for every edge $e \in E(G) - E_0$ there exists $e_0 \in E_0$ so that e and e_0 are incident to some common node $v_0 \in V(G)$.

A slightly different idea is that of a graph covering. Here $V_0 \subset V(G)$ is said to be a *covering* set of G if each $e \in E(G)$ is incident to some node $v_0 \in V_0$. The *covering number*, $\mathrm{Cov}(G)$, of G is then the cardinality of the smallest covering set of G. Similarly, $E_0 \subset E(G)$ is an *edge cover* of G if every $v \in V(G)$ is incident to some edge $e_0 \in E_0$. Note the duality between dominating sets and covering sets in that nodes dominate other nodes of G but cover edges. Similarly, edges dominate other edges but cover nodes of the graph [21].

A complementary concept is that of *independence* in a graph, which requires non-adjacency where domination insists on adjacency. Thus $V_0 \subset V(G)$ is an independent set of nodes if no two elements of V_0 are adjacent, and $E_0 \in E(G)$ is an independent edge set if no two edges in E_0 are incident to a common vertex $v_0 \in V(G)$. (An independent edge set is also called a *matching* of G.) The node- (resp. edge-) *independence* number, $\mathrm{Ind}(G)$, is then the cardinality of the largest set of independent nodes (resp. edges) in G. Clearly, the complement of a node cover of G is an independent set, and conversely.

Numerous relationships hold among these extreme values. If we use the subscripts e and v to distinguish between sets of edges and nodes of G, respectively, then the following statements are true [22]:

$$\mathrm{Dom}_e(G) \leq \mathrm{Ind}_e(G) \leq \mathrm{Cov}_v(G). \tag{7.15a}$$

$$\mathrm{Dom}_v(G) \leq \mathrm{Ind}_v(G) \leq \mathrm{Cov}_e(G). \tag{7.15b}$$

$$\mathrm{Ind}_e(G) + \mathrm{Cov}_e(G) = p, \tag{7.15c}$$

$$\mathrm{Ind}_v(G) + \mathrm{Cov}_v(G) = p. \tag{7.15d}$$

Also, as in the case of dominating sets, the notions of covering sets and independent sets can be further extended and quantified, and leverage sequences can be defined in terms of each one. These invariants have been used in applications to network scheduling problems, particularly those involved with parallel data processing. While they all have some relevence to network vulnerability in the broad sense, their more direct bearing on vulnerability and survivability requires much more investigation. Finally, some of these invariants again lead to computationally intractable problems. For example, given a graph $G = G(V, E)$ and an integer $k < |V(G)|$, the question of deciding whether there exists a vertex covering of G having cardinality at most k is known to be $\mathcal{NP}$-complete, and the same is true if we ask for an independent set of at most k vertices. This is not the case for independent edges, however, as an algorithm due to Edmonds [4,20] will find maximum matchings with complexity $O(p^4)$. In addition, some of the corresponding questions for stochastic graphs can be shown to be $\mathcal{NP}$-hard [167].

8. REFERENCES

8.1 GRAPH THEORY - GENERAL REFERENCES

The books cited here all discuss the various aspects of graph theory in general, not merely from a communication network point of view. The 1969 textbook by Harary has become a standard in this field, and we try to follow its notation as much as possible. Buckley and Harary contains a wealth of recently obtained material related to the hop distance and graph invariants defined in terms of distance properties, much of which may have applications to communications networks. Harary, Norman and Cartwright has a very complete discussion of connectivity in directed networks. The Capobianco and Molluzzo book presents a number of illuminative examples of relations holding among common graph invariants, such as node and edge connectivity, including counter examples which indicate the extent to which these results are best-possible.

[1] C. Berge, *Graphs*, North-Holland, New York, 1985.

[2] B. Bollobas, *Graph Theory: An Introductory Course*, Springer-Verlag, New York, 1979.

[3] F. Buckley and F. Harary, *Distance in Graphs*, Addison-Wesley, Reading, Massachusetts, 1990

[4] M. Capobianco and J. C. Molluzzo, *Examples and Counterexamples in Graph Theory*, North-Holland, New York, 1978

[5] W.-K. Chen, *Theory of Nets: Flows in Networks*, John Wiley and Sons, New York, 1990

[6] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice-Hall, Englewood Cliffs, New Jersey, 1974

[7] F. Harary, *Graph Theory*, Addison-Wesley, Reading, Massachusetts, 1969

[8] F. Harary, R. Z. Norman and D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs*, John Wiley and Sons, New York, 1965

[9] F. Harary and E. M. Palmer, *Graphical Enumeration*, Academic Press, New York, 1973

[10] D. F. Robinson and L. R. Foulds, *Digraphs : Theory and Techniques*, Gordon and Breach, New York, 1980

8.2 ALGORITHMS FOR GRAPHS AND NETWORKS

Graph-theoretic algorithms are given good coverage by Tarjan and in the two books

by Even, but Gibbons is probably the best introduction to this subject. Colbourn's book is the only general text available at this time which covers network reliability from the point of view of most interest to this study. McHugh includes a chapter on the implementation of graph algorithms for concurrent and parallel computation. Tenenbaum et al. discuss the implementation of some of these algorithms in the Pascal and C languages. See also [11, 15, 16] for more discussion of using data structures and graph algorithms in these computer languages. Christofides discusses some important graph algorithms in much more detail, including the travelling salesman and Hamiltonian tour problems. A number of these works also treat some topics involved with flows in networks, a subject which we have not discussed here but one which does have important applications to network connectivity, as discussed in the article by Estahanian and Hakimi.

[11] L. Ammeraal, *Programs and Data Structures in C*, John Wiley and Sons, New York, 1987

[12] W. Amsbury, *Data Structures: From Arrays to Priority Queues*, Wadsworth, Belmont, California, 1985

[13] N. Christofides, *Graph Theory: An Algorithmic Approach*, Academic Press, New York, 1975

[14] C. J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, London, 1987

[15] N. Dale and S. C. Lilly, *Pascal Plus Data Structures* (2nd edition), D. C. Heath, Lexington, Massachusetts, 1988

[16] J. Esakov and T. Weiss, *Data Structures: An Advanced approach Using C*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989

[17] A. H. Estahanian and S. L. Hakimi, "On Computing the Connectivity of Graphs and Digraphs", *Networks*, vol. 14 (1984), pp 355-366.

[18] S. Even, *Graph Algorithms*, Computer Science Press, Rockville, Maryland, 1979

[19] S. Even, *Algorithmic Combinatorics*, Macmillan, New York, 1973

[20] A. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, 1985

[21] J. A. McHugh, *Algorithmic Graph Theory*, Prentice Hall, Englewood Cliffs, New Jersey, 1990

[22] U. Manber, *Introduction to Algorithms*, Addison-Wesley, Reading, Massachusetts, 1989

[23] E. Minieka, *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, New York, 1978

[24] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982

[25] M. N. S. Swamy and K. Thulasiraman, *Graphs, Networks, and Algorithms*, John Wiley and Sons, New York, 1981

[26] R. E. Tarjan. Data *Structures and Network Algorithms*. SIAM. Philadelphia. Pennsylvania. 1983

[27] A. M. Tenenbaum and M. J. Augenstein. *Data Structures Using Pascal* (2nd edition). Prentice-Hall. Englewood Cliffs. New Jersey, 1986

[28] A. M. Tenenbaum. Y. Langsam. and M. J. Augenstein. *Data Structures Using C*. Prentice-Hall. Englewood Cliffs. New Jersey. 1990

8.3 COMPUTATIONAL COMPLEXITY

The references below all discuss the efficiency and complexity of computer algorithms in general. not merely the graph-theoretic algorithms. Here the two books by Aho et al. and Knuth's three volume series are the best general references to the theory and practice of computer algorithms. Garey and Johnson is the best overall guide to NP-completeness and provides a compendium of many of those problems that were known to be NP-complete as of 1979. There are now more than 1000 known NP-complete problems. many of them in graph theory, and dozens more are discovered every year. so this catalogue has rapidly become out of date. As a result there is now an ongoing column on NP-complete problems by Johnson which appears several times a year in the journal *Algorithms*. A number of these columns have discussed NP-completeness for problems in communication networks and reliability. Harel is a very good and exceptionally readable overall account of the current state of the art in algorithmics and has a good account of the problems that arise in designing and verifying algorithms for parallel processing. The book by Sedgewick also comes in two other editions which give more details of the implementations of these algorithms in either Pascal or in C.

[29] A. V. Aho. J. E. Hopcroft. and J. D. Ullman. *The Design and Analysis of Computer Programs*. Addison-Wesley. Reading. Massachusetts. 1974

[30] A. V. Aho. J. E. Hopcroft. and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley. Reading. Massachusetts, 1983

[31] T. H. Cormen. C. E. Leiserson. and R. L. Rivest. *Introduction to Algorithms*. MIT Press. Cambridge Massachusetts. 1990

[32] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. Freeman. San Francisco. California. 1979

[33] D. Harel. *Algorithmics: The Spirit of Computing*. Addison-Wesley. Reading. Massachusetts. 1987

[34] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms*. Computer Science Press. Rockville. Maryland. 1978

[35] D. S. Johnson, "The NP-Completeness Column: An Ongoing Guide", *Algorithms* (this column appears several times per year)

[36] D. E. Knuth, *The Art of Computing:* Addison-Wesley, Reading, MA
Vol 1: *Fundamental Algorithms* (2nd edition), 1973
Vol 2: *Seminumerical Algorithms* (2nd edition), 1981
Vol 3: *Sorting and Searching*, 1973

[37] R. Sedgewick, *Algorithms* (2nd edition), Addison-Wesley, 1988

[38] H. S. Wilf, *Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1986

8.4 *NP*-HARD AND *NP*-COMPLETE PROBLEMS

Many of the problems that are concerned with calculating network reliability are *NP*-hard or *NP*-complete, and the various articles by Ball or Provan are the best sources for this material. Johnson's columns on *NP*-complete problems discuss a number of these, while the paper of Valiant is concerned with the computational complexity of some counting problems in the theory of graphs and networks.

[39] E. Arikan, "Some Complexity Results about Packet Radio Networks", *IEEE Transactions on Information Theory*,Vol 30 (1984), pp 681-685.

[40] M. O. Ball, "Complexity of Network Reliability Computations", *Networks*, Vol 10, (1980), pp 153-165.

[41] M. O. Ball, "Computational Complexity of Network Reliability Analysis: An Overview", *IEEE Transactions on Reliability*, Vol 35, (1986), pp 230-239.

[42] W. Chen and N. Huang, "The Strongly Connecting Problem on Multihop Packet Radio Networks", *IEEE Transactions on Communications*, Vol 37, (1989), pp 293-295.

[43] D. S. Johnson, "The NP-Completeness Column: An Ongoing Guide", *Algorithms*, Vol 3, (1982), pp 182-195.
Algorithms, 5 (1984), pp 595-609.
Algorithms, 6 (1985), pp 145-159.
Algorithms, 6 (1985), pp 434-451.
Algorithms, 8 (1987), pp 438-448.

[44] J. S. Provan and M. O. Ball, "The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected", *SIAM Journal of Computation*,Vol 12 (1983), pp 777-788.

[45] J. S. Provan and M. O. Ball, "Computing Network Reliability in Time Polynomial in the Number of Cuts", *Operations Research*, Vol 32, (1984), pp 516-526.

[46] J. S. Provan, "The Complexity of Reliability Computations in Planar and Acyclic Graphs", *SIAM Journal of Computation*, Vol 12, (1983), pp 777-788.

[47] L. G. Valiant, "The Complexity of Enumeration and Reliability Problems", *SIAM Journal of Computing*, Vol 8, (1979), pp 410-421.

8.5 DEPENDENT FAILURE EVENTS

Problems concerning the statistical dependence of the network component failures are treated in the references below but are not discussed in any detail in this report. One should note that the assumption of independent failures can lead to either an overly pessimistic or an overly optimistic estimate of the true network reliability. The recent paper by Egeland and Huseby gives some results as to how one might determine which of these is the case.

[48] J. Y. Assous, "First- and Second-Order Bounds on Terminal Reliability", *Networks*, Vol 16, (1986), pp 319-329.

[49] T. Egeland and A. B. Huseby, "On Dependence and Reliability Computation", *Networks*, Vol 21, (1991), pp 521-546.

[50] H. Heffes and A. Kumar, "Incorporating Dependent Node Damage in Deterministic Connectivity Analysis and Synthesis of Networks", *Networks*, Vol 16, (1986), pp 51-65.

[51] Y. F. Lam and V. Li, "On Network Reliability Calculations with Dependent Failures", *Proceedings of the IEEE 1983 Global Telecommunications Conference* (GTC '83), San Diego, California, November, 1977, pp 1499-1503.

[52] Y. F. Lam and V. Li, "Reliability Modeling and Analysis of Communication Networks with Dependent Failures", *Proceedings IEEE INFOCOM*, 1985, pp 196-199.

[53] Y. F. Lam and V. Li, "Reliability Modeling and Analysis of Communication Networks with Dependent Failures", *IEEE Transactions on Communications*, Vol 34, (1986), pp 82-84.

[54] K. V. Lee and V. O. K. Li, "A Path-Based Approach for Analyzing Reliability of Systems with Dependent Failures and Multinode Components", *Proceedings IEEE INFOCOM*, 1990, pp 495-503.

[55] L. B. Page and J. E. Perry, "A Model for System Reliability with Common-Cause Failures", *IEEE Transactions on Reliability*,Vol 38, (1989), pp 406-410.

[56] E. Zemel, "Polynomial Algorithms for Estimation of Network Reliability", *Networks*, Vol 12, (1982), pp 439-452.

8.6 NETWORK RELIABILITY - SPECIAL CASES

Most of the probabilistic measures of network connectivity lead to computability problems that are *NP*-hard, so there has been considerable effort in searching for restricted classes of networks for which there are reliability algorithms with a smaller order of complexity. Thus the papers of Boesch and Pullen consider only constant probability of edge failures. This may reduce the problem to one in graph enumeration,

but this problem still has non-polynomial complexity. Similarly, the article by Bienstock considers only planar networks, and he proves the existence of an algoroithm whose complexity grows exponentially in the square root of p, rather than p itself. This is still very far from having polynomial growth, and to obtain that complexity even more drastic restrictions are necessary, as shown in the articles by Agrawal and Satayanarana and by Politof and Satyanarayana. Even very regular grid networks in the xy-plane yields NP-hard problems, as shown by the Clark and Colbourn article.

[57] A. Agrawal and A. Satayanarana, "An $O(|E|)$ Time Algorithm for Computing the Relaibility of a Class of Directed Networks", *Operations Research*, Vol 32 (1984), pp 493-515.

[58] D. Bienstock, "An Algorithm for Reliability Analysis of Planar Graphs", *Networks*, Vol 16, (1986), pp 411-422.

[59] F. Beichelt and P. Tittman, "A Generalized Reduction Method for the Connectedness Probability of Stochastic Networks", *IEEE Transactions on Reliability*, Vol 40, (1991), pp 198-204.

[60] F. T. Boesch, "On Unreliability Polynomials and Graph Connectivity in Reliable Network Synthesis", *Journal of Graph Theory*, Vol 10, (1988), pp 339-352.

[61] A. Bobbio andf A. Premoli, "Fast Algorithm for Unavailibility and Sensitivity Analysis of Series-Parallel Systems", *IEEE Transactions on Reliability*, Vol 31 (1982), pp 359-361.

[62] B. N. Clark and C. L. Colbourn, "Unit Disk Graphs", *Discrete Math.*, Vol 86 (1990), pp 165-177.

[63] C. L. Colbourn, "Network Resiliance", *SIAM Journal of Algebra and Discrete Math*, Vol 8, (1987), pp 404-409.

[64] W. H. Debany, P. K. Varshney, and C. R. P. Hartman, "Network Reliability Evaluation Using ProbabilityExpressions", *IEEE Transactions on Reliability*, Vol 35, (1986), pp 161-166.

[65] T. Politof and A. Satyanarayana, "A Linear-Time Algorithm to Compute the Reliability of Planar Cube-free Networks", *IEEE Transactions on Reliability*, Vol 39, (1990), pp 557-563.

[66] K. W. Pullen, "A Random Network Model of Message Transmission", *Networks*, Vol 16, (1986), pp 397-409.

[67] O. W. W. Yang, "Terminal Pair Reliability of Tree-Type Computer Communication Networks", *Proceedings of the IEEE 1991 Military Communications Conference (MILCOM '91)*, November, 1991.

8.7 SURVEY ARTICLES AND BIBLIOGRAPHIES

Some previous surveys have been devoted to different aspect of network reliability, the most recent of which seems to be that of Lam and Li, which dealt especially with

some problems of the statistical dependence of the network component failures. Also, a number of the earlier studies, such as the one by Boesch and those of Frank et al., were primarily concerned with applications of reliability calculations or estimates to the overall question of optimal network design. Brigham and Dutton give an exhaustive listing and cross-referencing of relations holding among the most common graph invariants, including the node and edge connectivities. The two surveys of Hedetniemi et al. contain quite extensive bibliographies of the current literature in several aspects of graph theory that are very relevant to communications networks, although not immediately applicable to reliability itself.

[68] A. Agrawal and R. E. Barlow, "A Survey of Network Reliability and Domination Theory", *Operations Research*, Vol 32, (1984), pp 478-492.

[69] F. T. Boesch, "Synthesis of Reliable Networks - A Survey", *IEEE Transactions on Reliability*, Vol 35, (1986), pp 240-246.

[70] R. C. Brigham and D. Dutton,
a. "Relations between Graph Invariants", *Networks*, Vol 15,(1985), pp 73-107.
b. "Supplement to 'Relations between Graph Invariants' ", *Networks*, Vol 21, (1991) pp 421-455.

[71] H. Frank and W. Chou, "Topological Optimization of Computer Networks", *Proceedings of the IEEE*, Vol 60, (1972), pp 1385-1387.

[72] H. Frank, "Survivability Analysis of Command and Control Communications Networks", *IEEE Transactions on Communications*, Vol 22, (1974),
Part I, pp 589-595.
Part II, pp 596-605.

[73] B. L. Golden and T. L. Magnanti, "Deterministic Network Optimization: A Bibliography", *Networks*, Vol 7, (1977), pp 149-183.

[74] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks", *Networks*, Vol 18, (1988), pp 319-349.

[75] S. T. Hedetniemi and R. C. Lasker, "Bibliography on Domination in Graphs and Some Basic Definitions of Domination Parameters", *Discrete Math.*, Vol 86 (1990), pp 257-277.

[76] C. L. Hwang, F. A. Tillman, and M. H. Lee, "System Reliability Evaluation Techniques for Complex Large Systems - A Review", *IEEE Transactions on Reliability*, Vol 30, (1981), pp 416-423.

[77] Y. F. Lam and V. Li, "A Survey of Network Reliability Modeling and Calculations", *Proceedings of the IEEE 1986 Military Communications Conference* (MILCOMM '86), Section 1.2.

[78] M. O. Locks, "Recursive Disjoint Products: A Review of Three Algorithms", *IEEE Transactions on Reliability*, Vol 31, (1982), pp 33-35.

[79] M. O. Locks, "Recent Developments in Computing of System Reliability", *IEEE Transactions on Reliability*, Vol 34, (1985), pp 425-436.

[80] A. Politof and A. Satyanarayana, "Efficient Algorithms for Reliability Analysis of Planar Networks - A Survey", *IEEE Transactions on Reliability*, Vol 35, (1986), pp 252-259.

[81] S. Rai and D. P. Agrawal (eds), "Distributed Computing Network Reliability", *IEEE Computer Press*, 1990.

[82] P. J. Slater, "A Summary of Results on Pair-Connected Reliability", *Contemporary Mathematics*, American Mathematical Society, 1989, pp 145-152.

8.8 PARALLEL GRAPH ALGORITHMS

The area of algorithms for parallel computation is a relatively new one that is the subject of intensive research and is not at all well understood at this time. The problems involved are all the more difficult because of the large number of computer architectures employed and proposed for concurrent and parallel machines. Indeed, some of these have been specifically designed with just one particular algorithm in mind, such as butterfly networks for FFT (fast Fourier transform) calculations or bitonic networks for fast parallel sorts. Nevertheless, numerous parallel implementations of the basic graph-theoretic algorithms have been proposed. The references cited here are provided merely to give some idea of what is currently being done along these lines.

[83] J. Bentley, "A Parallel Algorithm for Constructing Minimal Spanning Trees", *Journal of Algorithms*, Vol 1, (1980), pp 51-59.

[84] S. K. Das, N. Deo, and S. Prasad, "Parallel Graph Algorithms for Hypercube Computers", *Parallel Computing*, Vol 13, (1990), pp 143-158.

[85] S. K. Das, N. Deo, and S. Prasad, "Two Minimum Spanning Forest Algorithms Based on Fixed Size Hypercube Computers", *Parallel Computing*, Vol 15 (1990), pp 179-185.

[86] X. He, "Efficient Parallel Algorithms for Series-Parallel Graphs", *Journal of Algorithms*, Vol 12, (1991), pp 409-430.

[87] C. P. Kruskal, L. Rudolph, and M. Snir, "Efficient Parallel Algorithms for Graph Problems", *Algorithmica*, Vol 5, (1990), pp 43-64.

[88] M. J. Quinn and N. Deo, "Parallel Graph Algorithms", *ACM Computer Surveys*, Vol 16, (1984), pp 319-348.

[89] Y. Shiloach and U. Vishkin, "An $O(\log n)$ Parallel Connectivity Algorithm", *Journal of Algorithms*, Vol 3, (1982), pp 57-67.

8.9 STATE SPACE ENUMERATION

As mentioned in the text, state space enumeration is not feasable except for either

very small or extremely reliable networks. The two papers cited here describe an algorithm (ORDER) that is useful in the latter situation.

[90] Y. F. Lam and V. Li, "An Improved Algorithm for Performance Analysis of Networks with Unreliable Components", *IEEE Transactions on Communications*, Vol 34, (1986), pp 496-497.

[91] V. Li and J. A. Silvester, "Performance Analysis of Networks with Unreliable Components", *IEEE Transactions on Communications*, Vol 32,(1984), pp 1105-1110.

8.10 INCLUSION-EXCLUSION METHODS

Inclusion-exclusion methods were used at an early date by Cavers to estimate networks reliability, but as originally described, these were impractical for all but the very smallest networks. The seminal article by Satyanarayana and Prabhakar made this method practical for intermediate sized networks by greatly reducing the number of terms required in computing the network reliability by applying the inclusion-exclusion principle to the set of *st*-paths or *st*-cutsets. The papers by Shier and Whited give recent algorithms for the enumeration of *st*-cutsets, of which there are generally fewer than there are *st*-paths.

[92] J. A. Cavers, "Cutset Manipulations for Communication Network Reliability Estimation", *IEEE Transactions on Communications*, Vol 23, (1975), pp 569-575.

[93] S. P. Jain and K. Gopal, "An Efficient Algorithm for Computing Global Reliability of a Network", *IEEE Transactions on Reliability*, Vol 37, (1988), pp 488-492.

[94] M. O. Locks, "Recursive Disjoint Products, Inclusion-Exclusion, and Min-Cut Approximations", *IEEE Transactions on Reliability*, Vol 29, (1980), pp 386-371.

[95] A. Satyanarayana and A. Prabhakar, "New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex Networks", *IEEE Transactions on Reliability*, Vol 27, (1978), pp 23-32.

[96] A. Satyanarayana, "A Unified Formula for Analysis of Some Network Reliability Problems", *IEEE Transactions on Reliability*,Vol 31 (1982), pp 23-32.

[97] D. R. Shier and D. E. Whited, "Algorithms for Generating Minimal Cutsets by Inversion", *IEEE Transactions on Reliability*,Vol 34, (1985), pp 314-319.

[98] D. R. Shier and D. E. Whited, "Iterative Methods for Generating Minimal Cutsets in Directed Graphs", *Networks*, Vol 16, (1986), pp 133-147.

[99] D. R. Shier and D. E. Whited, "Algebraic Methods Applied to Network Reliability Problems", *SIAM Journal of Algebra and Discrete Math.*, Vol 1, (1987), pp 251-262.

[100] D. R. Shier, "Algebraic Methods for Bounding Network Reliability", *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol 5, (1991), pp 245-259.

8.11 DISJOINT SUMS AND BOOLEAN MINIMIZATION

That methods using disjoint sums were intimately related to the inclusion-exclusion methods was clear from the beginning. The improvements in the implementation of this method have been in the consistent application of Boolean analysis to reduce the number of terms required to represent the success and failure events for the network in question. The recent papers of Soh and Rai compares different ways of ordering the path or cutset events prior to accumulating the probabilities of the success or failure.

[101] J. A. Abraham, "An Improved Algorithm for Network Reliabilty", *IEEE Transactions on Reliability*, Vol 28, (1979), pp 58-61.

[102] M. O. Ball and J. S. Provan, "Disjoint Products and Efficient Computation of Reliability", *Operations Research*, Vol 36, (1988), pp 703-716.

[103] F. Beichelt and L. Spross, "An Improved Abraham-Method for Generating Disjoint Sums", *IEEE Transactions on Reliability*, Vol 36, (1987), pp 70-74.

[104] F. Beichelt and L. Spross, "Comment on 'An Improved Abraham-Method for Generating Disjoint Sums'", *IEEE Transactions on Reliability*, Vol 38 (1989), pp 422-424.

[105] F. Beichelt and L. Spross, "Bounds on the Reliability of Binary Coherent Systems", *IEEE Transactions on Reliability*, Vol 38, (1989), pp 425-427.

[106] S. Hariri and C. S. Raghavendra, "SYREL: A Symbolic Reliability Algorithm Based on Path and Cutset Methods", *IEEE Transactions on Computers*, Vol 36, (1987), pp 1224-1232.

[107] K. D. Heidtmann, "Smaller Sums of Disjoint Products by Subproduct Inversion", *IEEE Transactions on Reliability*, Vol 38, (1989), pp 305-311.

[108] R. Sahner and K. Trivedi, "Performance and Reliability Analysis Using Directed Acyclic Graphs", *IEEE Transactions on Software Engineering*, Vol 18, (1987), pp 1105-1114.

[109] S. Soh and S. Rai, "A Computer Approach for Reliability Evaluation of Telecommunication Networks with Heterogeneous Link-Capacities", *IEEE Transactions on Reliability*, Vol 40, (1991), pp 441-451.

[110] S. Soh and S. Rai, "Survivability Analysis of Complex Computer-Network with Heterogeneous Link-Capacities", *Proceedings Ann. Reliability and Maintainability Symposium*, (1991), pp 374-379.

[111] S. Soh and S. Rai, "CAREL: Computer Aided Reliability Evaluator for Distributed Computer Networks", *IEEE Transactions on Parallel and Distributed Systems*, Vol 2, (1991), pp 199-213.

[112] S. Soh and S. Rai, "Experimental Results on Preprocessing of Path/Cut Terms in Sums of Disjoint Product Technique", *Proceedings of the IEEE INFOCOMM 1991 Conference*, Bal Harbor, Florida, April 1991, pp 0533-0542.

[113] M. Veeraraghavan and K. Trivedi, "An Improved Algorithm for Symbolic Reliability Analysis", *IEEE Transactions on Reliability*, Vol 40, (1991), pp 347-358.

[114] J. M. Wilson, "An Improved Minimizing Algorithm for Sum of Disjoint Products", *IEEE Transactions on Reliability*, Vol 39, (1990), pp 42-45.

8.12 DOTSON'S METHOD

Dotson's method has been available for as long as most of the other methods, but it does not seem to have been used as much, in spite of the advantages it yields via upper and lower bounds on the network reliability. The 1988 paper of Yoo and Deo showed it to be quite competitive with the other algorithms tested therein, and that of Torrieri indicates how easy it is to adapt Dotson's approach to account for node failures (as well as edge failures) without any of the problems associated with the large increase in the size of the state space of the network.

[115] W. P. Dotson, "An Analysis and Optimization Technique for Probabilistic Graphs", *PhD Dissertation*, Air Force Institute of Technology, Wright Patterson AFB, August 1976 (DTIC access no AD-A028398; University Microfilms no 7709474).

[116] W. P. Dotson and J. O. Gobien, "A New Analysis Technique for Probabilistic Graphs", *IEEE Transactions on Circuits and Systems*, Vol 26, (1979), pp 855-865.

[117] J. S. Lee Associates, "Development of Analysis Models for MSE Network Performance in the Tactical Environment",
Vol 1: *Automated Methods for Calculating Connectivity and Link Flow*
Vol 2: *Implementation of an Equivalent-Links Concept*
Vol 3: *Techniques for Analyzing Large Networks*

[118] D. Torrieri, "An Efficient Algorithm for the Calculation of Node-Pair Reliability", *Proceedings of the IEEE 1991 Military Communications Conference* (MILCOM '91), November, 1991.

[119] Y. B. Yoo and N. Deo, "A Comparison of Algorithms for Terminal Pair Reliability", *IEEE Transactions on Reliability*,Vol 37,(1988), pp 210-215.

8.13 FACTORIZATION METHODS

Factorization methods have long been considered as very attractive candidates for reliability algorithms because of their relatively small storage requirements, but have

been efficiently implemented only recently, as best described in a series of papers by Page and Perry. The important point here seems to be the choice of edge reductions to apply at each point in the recursive function call. The article by Johnson gives some comparisons on the size of the recursion trees generated by calls to the edge factorizations for various choices of the edge reductions that are to be performed prior to each factorization.

[120] T. A. Feo and R. Johnson, "Partial Factoring: An Efficient Algorithm for Approximating 2-Terminal Reliability on Complete Graphs", *IEEE Transactions on Reliability*, Vol 39, (1990), pp 290-295.

[121] R. Johnson, "Network Reliability and Acyclic Orientations", *Networks*, Vol 14 (1984), pp 489-505.

[122] L. B. Page and J. E. Perry, "A Practical Implementation of the Factor Theorem for Network Reliability", *IEEE Transactions on Reliability*, Vol 37, (1988), pp 259-267.

[123] L. B. Page and J. E. Perry, "Reliability of Directed Networks using the Factor Theorem", *IEEE Transactions on Reliability*, Vol 38, (1989), pp 556-562.

[124] A. Satyanarayana and M. K. Chang, "Network Reliability and the Factoring Theorem", *Networks*, Vol 20, (1983), pp 107-120.

[125] O. R. Theologou and J. G. Carlier, "Factoring & Reduction for Networks with Imperfect Vertices", *IEEE Transactions on Reliability*, Vol 40, (1991), pp 210-217.

[126] R. K. Wood, "Factoring Algorithms for Computing K-terminal Network Reliability", *IEEE Transactions on Reliability*, Vol 35, (1986), pp 269-278.

8.14 BOUNDS AND APPROXIMATIONS

In view of the computational intractability of the network reliability calculations for general probabilistic graphs, there are a very large number of papers devoted to the problem of obtaining accurate estimates of the reliability. Early versions of these assumed a constant probability of edge failure and were intent on computing the coefficients of the corresponding reliability polynomial, but these coefficient problems are equivalent to some graph enumeration problems which were also shown to be intractable. More recent methods apply to networks with arbitrary failure probabilities, and the edge-packing approach of Colbourn et al. is of most interest here.

[127] H. M. AboElFotoh and C. J. Colbourn, "Series-Parallel Bounds for the Two-Terminal Reliability Problem", *ORSA Journal on Computing*, Vol 1, (1989), pp 209-222.

[128] H. M. AboElFotoh and C. J. Colbourn, "Computing 2-Terminal Reliability for Radio-Broadcast Networks", *IEEE Transactions on Reliability*, Vol 38, (1989), pp 538-555.

[129] M. O. Ball and J. S. Provan, "Calculating Bounds on Reachability and Connectedness in Stochastic Networks", *Networks*, Vol 13, (1983), pp 253-278.

[130] M. O. Ball and J. S. Provan, "Bounds on the Reliability Polynomial for Shellable Independence Systems", *SIAM Journal of Algebra and Discrete Math*, Vol 3 (1981), pp 166-181.

[131] T. B. Brecht and C. J. Colbourn, "Improving Reliability Bounds in Computer Networks", *Networks*, Vol 16, (1986), pp 369-380.

[132] T. B. Brecht and C. J. Colbourn, "Multplicative Improvements in Network Reliability Bounds", *Networks*, Vol 19, (1989), pp 521-529.

[133] T. B. Brecht and C. L. Colbourn, "Lower Bounds on Two-Terminal Network Reliability", *Discrete Applied Math*, Vol 21, (1988), pp 185-198.

[134] C. J. Colbourn and A. Ramanathan, "Bounds for All-Terminal Reliability by Arc-Packing", *Ars Combinatorica*, Vol 23A, (1987), pp 229-236.

[135] C. J. Colbourn, "Edge-Packings of Graphs and Network Reliability", *Discrete Math*, Vol 72, (1988), pp 49-61.

[136] C. J. Colbourn and D. D. Harms, "Bounding All-Terminal Reliability in Computer Networks", *Networks*, Vol 18, (1988), pp 1-12.

[137] D. Torrieri, "Algorithms for Finding an Optimal Set of Short Disjoint Paths in a Communication Network", *Proceedings of the IEEE 1991 Military Communications Conference (MILCOM '91)*, November, 1991.

[138] D. K. Wagner, "Disjoint (s,t)-Cuts in a Network", *Networks*, Vol 20, (1990), pp 361-371.

8.15 SIMULATIONS AND MONTE CARLO METHODS

Studies of network performance have often used Monte Carlo simulations to approximate various network parameters and performance measures, the main problem being the large number of sample trials necessary to obtain reliable estimates. This approach is not discussed in our study, but there are a number of recent references to this topic that should be noticed. Moreover, it is possible to model some statistically dependent component failures with these methods.

[139] G.S. Fishman, "A Comparison of Four Monte Carlo Methods for Estimating the Probability of *st*-connectedness", *IEEE Transactions on Reliability*, Vol 35 (1986), pp 145-154.

[140] G.S. Fishman, "A Monte Carlo Sampling Plan for Estimating Network Reliability", *Operations Research*, Vol 34, (1986), pp 581-594.

[141] G.S. Fishman, "Estimating the *st*-Reliability Function Using Importance and Stratified Sampling", *Operations Research*, Vol 37, (1989), pp 462-473.

[142] G.S. Fishman, "A Monte Carlo Sampling Plan for Estimating Reliability Parameters and Related Functions", *Networks*, Vol 17, (1987), pp 169-186.

[143] R. M. Karp and M. G. Luby, "Monte-Carlo Algorithms for Enumeration and Reliability Problems", *Proc IEEE 24th Annual Symposium on Foundations of Computer Science*, November 7-9, Tuscon, Arizona, pp 56-64.

[144] P. Kubat, "Estimation of Reliability for Communication/Computer Networks - Simulation/Analytic Approach", *IEEE Transactions on Communications*, Vol 37, (1989), pp 927-933.

[145] L. D. Nel and C. J. Colbourn, "Combining Monte Carlo Estimates and Bounds for Network Reliability", *Networks*, Vol 20, (1990), pp 277-298.

8.16 CONNECTIVITY FACTORS

The NCF (node connectivity factor) and the LCF (link connectivity factor) were introduced in the papers cited below as possible alternatives to the usual reliability measure. They are indicators of how close the network is to being totally disconnected. Unfortunately, the NCF at least is computationally difficult to compute and does not seem to be amenable to simplifying techniques such as factorization or edge reduction used by other methods. Thus it is not yet clear how useful a concept this will prove to be, although if these connectivity factors are available they can be used to identify the most vulnerable components of the network and to adapt the network so as to equalize the vulnerability over its components.

[146] K. T. Newport and M. A. Schroeder, "Network Survivability through Connectivity Optimization", *Proceedingsof the 1987 IEEE International Conference on Communications*, Vol 1, pp 471-477.

[147] K. T. Newport and P. Varshney, "Design of Communications Networks Under Performance Constraints", *IEEE Transactions on Reliability*, Vol 40, (1991), pp 443-439.

[148] K. T. Newport and M. A. Schroeder, "Techniques for Evaluating the Nodal Survivability of Large Networks", *Proceedings of the IEEE 1990 Military Communications Conference* (MILCOM '90), Monterey, California, pp 1108-1113.

[149] K. T. Newport, M. A. Schroeder, and G. M. Whittaker, "A Knowledge Based Approach to the Computation of Network Nodal Survivability", *Proceedings of the IEEE 1990 Military Communications Conference* (MILCOM '90), Monterey, California, pp 1114-1119.

[150] M. A. Schroeder and K. T. Newport, "Tactical Network Survivability through Connectivity Optimization", *Proceedingsof the 1987 Military Communications Conference* (MILCOM '87), Monterey, California, Vol 2, pp 590-597.

[151] G. M. Whittaker, "A Knowledge-Based Design Aid for Survivable Tactical Networks", *Proceedings of the IEEE 1990 Military Communications Conference* (MILCOM '90), Monterey, California, Sect 53.5.

[152] M. A. Schroeder and K. T. Newport, "Enhanced Network Survivability Through Balanced Resource Criticality", *Proceedings of the IEEE 1989 Military Communications Conference* (MILCOM '89), Boston, Massachusetts, Sect 38.4.

8.17 GRAPH DIAMETER AND CONNECTIVITY

The papers cited below are concerned with some aspects of graph connectivity other than the usual path-oriented one, primarily with those deriving from the notion of the diameter of a graph (i.e., the maximum node-to-node hop distance across the graph) or the average node-to-node hop distance. Of special interest here is the notion of leverage, as described in the papers of Bagga et al., which is a general method of quantifying changes in graph invariants due to the loss of some network components.

[153] K. S. Bagga, L. W. Beineke, M. J. Lipman, R. E. Pippert, "The Concept of Leverage in Network Vulnerability", *Conference on Graph Theory*, Kalamazoo, Michigan, (1988), pp 29-39.

[154] K. S. Bagga, L. W. Beineke, M. J. Lipman, R. E. Pippert, "Explorations into Graph Vulnerability", *Conference on Graph Theory*, Kalamazoo, Michigan, (1988), pp 143-158.

[155] D. Bienstock and E. Gyori, "Average Distance in Graphs with Removed Elements", *Journal of Graph Theory*, Vol 12, (1988), pp 375-390.

[156] F. T. Boesch and I. T. Frisch, "On the Smallest Disconnecting Set in a Graph", *IEEE Transactions on Circuit Theory*, Vol 15, (1986), pp 286-288.

[157] F. Buckley and M. Lewinter, "A Note on Graphs with Diameter preserving Spanning Trees", *Journal of Graph Theory*, Vol 12, (1988), pp 525-528.

[158] F. R. K. Chung, "The Average Distance and the Independence Number", *Journal of Graph Theory*, Vol 12, (1988), pp 229-235.

[159] G. Exoo, "On a Measure of Communication Network Vulnerability", *Networks*, Vol 12, (1982), pp 405-409.

[160] O. Favaron, M. Kouider, and M. Makeo, "Edge-Vulnerability and Mean Distance", *Networks*, Vol 19, (1989), pp 493-509.

[161] F. Harary, F. T. Boesch, and J. A. Kabell, "Graphs as Models of Communication Network Vulnerability: Connectivity and Persistence", *Networks*, Vol 11 (1981), pp 57-63.

[162] F. Harary, "Conditional Connectivity", *Networks*, Vol 13, (1983), pp 347-357.

[163] S. M. Lee, "Design of ϵ-invariant Networks", *Congressus Numer.*, Vol 65 (1988), pp 105-102.

[164] O. R. Oellermann, "Conditional Graph Connectivity Relative to Hereditary Properties", *Networks*, Vol 21, (1991), pp 245-255.

[165] J. Plesnik, "On the Sum of All Distances in a Graph or Digraph", *Journal of Graph Theory*, Vol 8, (1984), pp 1-21.

[166] A. A. Schoone, H. L. Bodlaender, and J. van Leeuwer, "Diameter Increase Caused by Edge Deletion", *Journal of Graph Theory*, Vol 11, (1987), pp 409-427.

8.18 OTHER MEASURES OF VULNERABILITY

The remaining references are concerned with a number of other graph invariants that have an obvious connection with the notions of vulnerability and survivability of communications networks. The main concepts here are those of dominance, independence, and covering of a graph with respect to either a set of nodes or a set of edges of the underlying graph. These quantities have already been applied to problems involving networks used in scheduling and service facilities, though their applications and usefulness to communications networks remains to be determined. Also, the calculation of some of these quantities can be NP-hard (some in the deterministic sense, others from the probabilistic point of view). This is also an area of very active research.

[167] M. O. Ball, J. S. Provan, and D. R. Shier, "Reliability Covering Problems", *Networks*, Vol 21, (1991), pp 345-357.

[168] L. Caccetta, "Vulnerability in Communication Networks", *Networks*, Vol 14 (1984), pp 141-146.

[169] L. L. Doty, "Extremal Connectivity and Vulnerability in Graphs", *Networks*, Vol 19, (1989), pp 73-78.

[170] T. J. Ferguson, J. H. Cozzens, and C. Cho, "SDI Network Connectivity Optimization", *Proceedings of the IEEE 1990 Military Communications Conference* (MILCOM '90), Monterey, California, Sect 53.1.

[171] J. F. Fink, M. S. Jacobson, L. F. Kinch, and J. Roberts, "The Bondage Number of a Graph", *Discrete Math.*, Vol 86, (1990), pp 47-57.

[172] G. Gunther, "Neighbor-Connectedness in Regular Graphs", *Discrete Applied Math.*, Vol 11, (1985), pp 233-242.

[173] G. Gunther, B. L. Hartnell, and R. Nowakowski, "Neighbor-Connected Graphs and Projective Planes", *Networks*, Vol 17, (1987), pp 241-247.

[174] P. L. Hammer, "Cut-threshold Graphs", *Disrete Applied Math.*, Vol 30, (1991), pp 163-179.

[175] A. M. Hobbs, "Computing Edge-Toughness and Fractional Arboricity", *Contemporary Mathematics, American Mathematical Society*, (1989), pp 89-106.

[176] T. Z. Jiang, "A New Definition on Survivability of Communication Networks", *Proceedings of the IEEE 1991 Military Communications Conference (MILCOM'91)*, November, 1991.

[177] L. M. Lesniak and R. E. Pippert, "On the Edge-Connectivity Vector of a Graph", *Networks*, Vol 19, (1989), pp 667-671.

[178] Z. Miller and D. Pritikin, "On the Separation Number of a Graph", *Networks*, Vol 19, (1989), pp 651-666.

[179] L. Wu and P. K. Varshney, "On Survivability Measures for Military Networks", *Proceedings of the IEEE 1990 Military Communications Conference* (MILCOM '90), Monterey, California, pp 1120-1124.